НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інституту прикладного системного аналізу
Кафедра математичних методів системного аналізу

До захисту допущено:
Завідувач кафедри
_____ Оксана ТИМОЩУК
«___»_____2024 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системний аналіз і управління»

спеціальності 124 «Системний аналіз»

на тему: «Аналіз впливу новин на курс біткоїна, класифікація їхнього впливу»

Виконала:
студентка IV курсу, групи КА-01
Магаріна Анна В'ячеславівна _____

Керівник:
Асистент кафедри ММСА,
Канцедал Георгій Олегович _____

Консультант з економічного розділу:
Доцент, к. е. н.
Рощина Надія Василівна _____

Рецензент:
Доцент, к. ф.-м. н.
Хорольський Олексій Вікторович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2024

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 124 «Системний аналіз»
Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Оксана ТИМОЩУК
«___»_____2024 р.

ЗАВДАННЯ
на дипломну роботу студентці
Магаріній Анні В'ячеславівни

1. Тема роботи «Аналіз впливу новин на курс біткоїна, класифікація їхнього впливу», керівник роботи Канцедал Георгій Олегович, асистент кафедри ММСА, затверджені наказом по університету від «___»_____ 2024 р. №_____

2. Термін подання студентом роботи: 11.06.2024 р.

3. Вихідні дані до роботи: зібрані реальні новини та показники курсу біткоїну на фінансовому ринку через певні проміжки часу в форматі датасету.

4. Зміст роботи: дослідження предметної області; опис теорії, що необхідна для реалізації програмного продукту; програмна реалізація та аналіз результатів; функціонально-вартісний аналіз програмного продукту.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): ілюстрації існуючих прогрманих веб-продуктів, ілюстрації до теоретичної частини, таблиці порівняння результатів прогнозів, графіки демонстрації роботи

програмного продукту, таблиці та графіки до функціонально-вартісного аналізу, презентація.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Н.В., доцент, к.е.н.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вибір теми і постановка дослідження	20.04.2024	Виконано
2.	Аналіз актуальності задачі дослідження	27.04.2024	Виконано
3.	Збір та аналіз літератури	02.05.2024	Виконано
4.	Формулювання задачі дослідження	10.05.2024	Виконано
5.	Розробка програмного продукту	03.05.2024	Виконано
6.	Аналіз та впорядкування результатів	18.05.2024	Виконано
7.	Оформлення пояснювальної записки	26.05.2024	Виконано
8.	Оформлення презентації для демонстрації	02.06.2024	Виконано

Студентка Анна МАГАРІНА

Керівник Георгій КАНЦЕДАЛ

РЕФЕРАТ

Дипломна робота: 92 с., 31 рис., 8 табл., 2 додатки, 17 джерел.

ПЕРЕДНАВЧЕНІ МОДЕЛІ, ЛІНГВІСТИЧНІ МОДЕЛІ, ПРОГНОЗУВАННЯ КУРСУ КРИПТОВАЛЮТИ, НЕЙРОННІ МЕРЕЖІ, ФІНАНСОВІ РИНКИ, БІТКОЇН.

Дана робота присвячена дослідженню впливу новин на курс криптовалюти, розробці методу прогнозування коливання курсу криптовалюти на основі новин. У наші часи на різні процеси, у тому числі фінансові, впливає велика кількість факторів, що веде до високої динамічності. Курси різних валют, особливо криптовалют, можуть зазнавати коливань і швидко змінюватися через вплив актуальної ситуації у світі, висвітленої через новини. За результатами роботи очікується дослідити вплив новин на коливання курсу криптовалюти та розглянути, наскільки можливо розробити модель для прогнозу на основі цих новин. Вихідні дані включають провідні новини та відповідно ціни та інші характеристики біткоїна.

Об’єкт дослідження – аналіз впливу новин на коливання ринку криптовалют та підходи до прогнозування курсу.

Мета роботи – розробити модель прогнозування курсу криптовалюти на основі провідних новин, проаналізувати вплив новин на курс криптовалют.

Результати роботи – було створено і проаналізовано розроблений підхід до прогнозування курсу криптовалюти на основі новин природною мовою. Було оцінено ефективність побудованого прогнозу за обраними метриками. Також проаналізовано вплив новин на коливання курсу криптовалюти біткоїн та ефективність підходів до прогнозу за допомогою лінгвістичної моделі.

Програмний продукт було створено за допомогою мови програмування Python.

ABSTRACT

The diploma work contains: 92 pages, 31 figures, 8 tables, 2 appendices, 17 references.

PRE-TRAINED MODELS, LINGUISTIC MODELS, CRYPTOCURRENCY PRICE PREDICTION, NEURAL NETWORKS, FINANCIAL MARKETS, BITCOIN.

This work is dedicated to studying the impact of news on cryptocurrency prices and developing a method for predicting cryptocurrency price fluctuations based on news. Nowadays, many factors influence various processes, including financial ones, leading to high dynamism. The rates of different currencies, especially cryptocurrencies, can fluctuate and change rapidly due to the influence of current global events, as covered by the news. The study aims to investigate the impact of news on cryptocurrency price fluctuations and examine the feasibility of developing a prediction model based on this news. The source data includes leading news articles, corresponding prices and other characteristics of Bitcoin.

Research object – analysis of the impact of news on cryptocurrency market fluctuations and approaches to price prediction.

The purpose of the work – to develop a model for predicting cryptocurrency prices based on leading news and to analyze the impact of news on cryptocurrency prices.

Results of the work – a developed approach to predicting cryptocurrency prices based on news in natural language was created and analyzed. The effectiveness of the constructed prediction was assessed according to selected metrics. The impact of news on Bitcoin price fluctuations and the effectiveness of the prediction approaches using a linguistic model were also analyzed.

The software product was created using Python.

ЗМІСТ

ВСТУП

Останніми роками, внаслідок комп’ютеризації, широкого поширення набули криптовалюти як альтернатива класичним грошовим валютам. Вони зайняли далеко не останнє місце в економічній сфері, маючи значний вплив на неї. Криптовалюти стали широко використовуватись як предмет інвестування, розрахункова одиниця тощо. Криптовалюти не мають внутрішньої вартості, тому їхні ціни можуть коливатися під впливом інформаційних збурень. Така віртуальна валюта має купу переваг, таких як повна анонімність, доступність по всьому світу у будь-який час, надійність операцій та інше, тому і користується популярністю по всьому світу.

Криптовалютні операції займають значну частину фінансового ринку. Оскільки інвестування в криптовалюту стало актуальним як для крупних інвесторів, так і для окремих звичайних людей, з’явилась нагальна потреба у вивчені впливів на коливання та передбаченні коливань курсу криптовалют.

Задача прогнозування курсу криптовалют є доволі складною та комплексною, оскільки на зміни курсу можуть впливати багато різних факторів. Цей процес потребує постійного аналізу великої кількості інформації, тому виникає потреба в певній автоматизації цього процесу.

Візьмемо до уваги, що пересічній людині, яка не має глибоких знань у криптовалютній сфері, доволі складно розібратися, як керувати цим ресурсом, навіть маючи значні фінансові можливості для капіталовкладень, як приклад. Ось тут і з’являється економічна вигода створення та реалізації продукту як для крупних компаній, так і для окремих користувачів, що може прогнозувати курс криптовалюти на основі певних даних.

Новини є збірним джерелом інформації, що висвітлює різні події в абсолютно всіх сферах, які можуть мати вплив на ринок криптовалют. Постає питання імплементації підходу, який може аналізувати актуальні новини та робити прогноз із урахуванням цих даних.

Наразі існують різні методи та підходи до прогнозування курсу криптовалют.

З розвитком штучного інтелекту та великих мовних моделей, з’явилась можливість обробляти великі обсяги інформації, різними моделями, навіть адаптуючи інформацію, представлену звичайними людськими мовами, для розуміння тими самими моделями. Але існує потреба пошуку найкращих рішень та методик задля збільшення точності прогнозу.

Метою цієї роботи є побудова моделі прогнозування курсу біткоїна на основі новин за допомогою лінгвістичної моделі, а саме із застосуванням двоспрямовані кодувальні представлення з трансформерів.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Історія створення криптовалют

Розвиток інформаційних технологій та діджиталізація сприяли виникненню нових різновидів цифрової валюти. Переваги таких валют є децентралізованість і свобода здійснення транзакцій.

Перші спроби створення криптовалюти були здійснені в 1990-х роках, але так і не змогли набути широкого поширення та закріпитися. У 1983 році співробітник факультету обчислювальної техніки Каліфорнійського університету в Санта-Барбарі Девід Чаум (англ. David Chaum) мав спробу розробки системи, що дозволяє одночасно забезпечити анонімність платежів із прозорістю для кожної сторони ринку. Було запропоновано використання «сліпого підпису», який надавав можливість проведення угоди між двома анонімними користувачами, але забезпечував видимість самого факту здійснення угодини стороннім спостерігачам. Чаум розробив протоколи «електронної готівки» (англ. digital cash) спільно з його ізраїльськими колегами. Це відкрило шлях до здійснення електронних платежів аналогічно до оплати готівковими коштами без посередників [1].

У 1997 році британець Адам Бек (англ. Adam Back) винайшов прообраз механізму створення криптовалюти. Його пропозицією було використання системи захисту від спаму Hashcash. Ця система передбачає те, що відправник робить багато тривалих транзакцій, а одержувач може швидко перевірити їхню справжність [1].

У 1998 році дослідник Нік Сабо розпочав роботу над децентралізованою грошовою системою під назвою «Bit gold». На його думку, електронні гроші можуть ефективно захищати від крадіжок, підробок і навіть інфляції. Проте, «Bit gold» не була анонімною, оскільки Сабо не бачив в цьому необхідності. Він прагнув створити віртуальні гроші, які потребували б певних зусиль для їхнього видобутку.

Для цього учасникам фінансової системи пропонувалося вирішувати криптографічні рівняння за допомогою комп'ютера, щоб заробити віргуальні гроші. Отримані відповіді надсилалися іншим учасникам для перевірки, і якщо вони визнавали їх правильними, то авторство результатів підтверджувалося. Ці відповіді ставали основою для наступних завдань, що призводило до збільшення ланцюжка грошової маси в обігу [1].

Однак Нік Сабо не зміг вирішити проблему подвійних витрат, коли власник «Bit gold» міг скопіювати відповідь на рівняння і використати одні й ті самі гроші двічі. Було запропоновано створити центральний контролюючий орган для ведення запису операцій, але ця ідея не подобалася Сабо, оскільки вона усувала основну перевагу системи — децентралізацію. Також залишилися невирішеними питання номіналу грошей, оцінки трудовитрат на їхній видобуток, віри людей у цінність віртуальних грошей і забезпечення контролю транзакцій. Сабо продовжував працювати над своїм проєктом до 2005 року, стикаючись із цими викликами. Проте він не зміг втілити його у життя [1].

Біткоїн став першою успішною реалізацією концепції криптовалюти. Принцип роботи біткоїну було описано ще у 2008 році [2]. Його засновником вважають Сатоші Накамото (англ. Satoshi Nakamoto). Він спромігся описати технічні аспекти функціонування майбутньої криптовалюти ще наприкінці жовтня 2008, а трохи пізніше, у січні 2009 року завантажив у мережу першу версію коду цієї електронної валюти. Йому вдалося успішно використати напрацювання своїх попередників і створити валюту, що надавала можливість здійснювати повністю анонімні транзакції [1].

Перша тестова транзакція, зроблена за допомогою криптовалюти, була здійснена в системі біткоїн 12 січня 2009 року. Сатоші Накамото зробив відправку 10 біткоїнів комп’ютерному спеціалісту Хелу Фінні (англ. Hal Finney). Ця транзакція відома під назвою «блок 170» (номер блоку, у якому вона була зафіксована) [1].

Майже до року біткоїн нічого фактично не коштував. Відбулась невелика кількість тестових транзакцій програмістами, що зацікавилися цією

криптовалютою. Ніхто нічого не купував та не оплачував цією криптовалютою. Не існувало ринкових обмінних курсів з іншими грошовими валютами.

Перша реальна успішна транзакція за допомогою біткоїна відбулась на початку жовтня 2009 року. Користувач під іменем New Liberty Standard оцінив вартість 1 309,03 біткоїна приблизно в 1 долар. За тиждень він придбав у Марті Мальмі 5050 біткоїнів за 5.05 доларів через платіжну систему «PayPal». Тобто ціна біткоїна була 0.001 долара. Надалі все більше користувачів цікавилось обміном традиційних грошових валют на біткоїн [3].

Перший обмін криптовалюти на фізичний товар відбувся 22 травня 2010 року у Флориді. Програміст Ласло Ханеч (англ. Laszlo Hanyecz) обміняв 10 000 біткоінів на 2 піци. Уже 3 грудня 2013 року 1 біткоїн коштував 1 078 доларів [1].

Журнал «Forbes» у 2011 році вперше опублікував статтю про систему біткоїн, де її було названо «криптовалютою» (англ. «crypto currency»), що популяризувало використання такого терміна. Більшість розробників (також і Сатоші Накамото) до того звикли використовувати термін «електронна готівка» (англ. «digital cash») [1].

1.2 Особливості та переваги операцій із криптовалютами

Традиційні механізми оплати використовують централізований підхід. Але для обробки транзакції біткоїнів не потрібна третя особа. Використовується технологія блокчейну, яка забезпечує новий підхід обробки платежів.

Блокчейн – це децентралізована база даних, яка забезпечує валідацію даних і дає змогу передавати інформацію в режимі реального часу. Учасники такої мережі мають різний рівень доступу до інформації. Ця мережа фактично є цифровим регістром або базаою даних, де вся інформація зберігається у вигляді блоків, при тому кожен наступний блок містить в собі зашифровану інформацію про попередні блоки [4].

Отже, це ланцюг «блоків», які представляють собою певну операцію в спеціально розподіленому регістрі. Технологія базується на процесі хешування (використовується спеціальна хеш-функція, яка перетворює вхідні дані в рядок із 32 символів – «хеш». Це надає високий рівень безпеки, адже кожна наступна транзакція хешується на основі попередніх. Будь-яка спроба підробки даних потребує значної кількості складних математичних обрахунків. Додатковим фактором безпеки є децентралізація та прозорість, за рахунок розподіленого характеру бази даних, що дозволяє проводити контроль безпосередньо самими користувачами [5].

1.3 Вплив криптовалюти на міжнародну економіку

Криптовалюти вже сьогодні мають суттєвий вплив на міжнародну економіку, сила якого зростатиме. Розглянемо основні перспективи такого впливу, які були згадані на конференції із цього питання [6].

1. Формування нової індустрії. Не зважаючи на недовгу історію існування криптовалют, вже створена ціла індустрія, побудована навколо них. До неї входять інституції, які здійснюють нагляд за операціями з цифровими монетами по всьому світу. Індустрія криптовалют швидко розвивається. Біткоїн, найвідоміша з цих криптовалют, уже дозволила багатьом людям та компаніям примножити власні статки, активно долучаючись до торгівлі криптовалютами. Міжнародна економіка покроково адаптується до цих змін, визнаючи, що криптовалюти мають значний потенціал.

2. Нові фінансові можливості для країн з недосконалою банківською системою, у якості альтернативи банкам. Велика частина населення світу не має легкого доступу навіть до базових банківських послуг. Наразі існує велика кількість додатків і програм, що можуть спростити та покращити досвід з використання криптовалют. Таким чином вони наближають їх до широкої

аудиторії. Торгівля криптовалютами може здійснюватися вільно через кордони, завдячуючи повній децентралізація процесів. Новітні технології із цифровими валюатми сприятимуть початку фінансової революції, яка відкриє нові можливості для суб'єктів цього процесу.

3. Зниження транзакційних витрат. Завдяки тому, що криптовалюти та блокчейн не потребують фізичних приміщень для виготовлення та обміну, витрати, які пов'язані з їхніми транзакціями, є мінімальними. Відсутня необхідність виплачувати заробітну плату, оренду та комунальні послуги, тому ці малі затрати забезпечують нижчі комісійні стягнення за операції. Це спонукає все більше людей до корисутвання цим новим фінансовим інструментом.

4. Прозорість транзакцій. Усі транзакції блокчейну та криптовалют повністю відстежуються, адже вони є автоматизованими та оцифрованими. Це забезпечує зниження ймовірності маніпулювання криптовалютами, що призводить до мінімізації ризиків шахрайства та корупції. Фактично слаборозвинені країни завдяки цьому підвищують свої шанси вступити у гру фінансових операцій та посилити власну економіку та соціальні перспективи. Щобільше, громадяни матимуть можливість відстежувати операції з державними коштами, що є актуальним для нашої країни далеко не в останню чергу, і таким чином матимуть змогу впливати на процеси у своєму політичному кліматі.

5. Збільшення можливостей для підприємців. Технологія блокчейн та криптовалюти можуть допомогти підприємцям розширити список доступних валют, у яких можливо отримувати платежі.

Отже, розвиток ринку криптовалют сильно впливає на різні сфери життя людей. Усе активніше з'являється тенденція до використання віртуальних грошових одиниць, про переваги яких було згадано вище. Технологія криптовалют ще є достатньо новою та незвичною широкому колу людей, тому потребується час для активного запровадження більшістю країн світу на рівні із традиційними валютами. Криптовалюти здійснили певну революції у фінансовій системі. Людству ще роками доведеться вивчати та впроваджувати цю технологію в маси, зіштовхуючись із достатньою кількістю викликів, таких як, наприклад, правова

сторона питання. Але вже не можливо заперечувати, що криптовалюти мають позитивне майбутнє.

1.4 Аналіз існуючих рішень прогнозування курсу криптовалюти

Розглянемо деякі відомі ресурси, які надають прогноз криптовалют. Нижче наведено програмні продукти, які є досить популярними та рекомендовані для використання у великій кількості статей, одну з яких [7] було використано для вивчення декількох з них.

«CryptoPredictions» використовує ШІ, машинне навчання та складні алгоритми для аналізу історичних даних з криптовалютних бірж. Його головна мета – допомогти трейдерам у виявленні перспективних нових проєктів у галузі криптовалют.

Інструмент пропонує безкоштовні прогнози цін для понад 19 000 різних криптовалют. Його перевагою є надання оновлених даних кожні 5 хвилин для криптовалют з значним ринковим капіталом. Крім того, «CryptoPredictions» обладнана корисними інструментами, такими як фільтри для криптовалют, автоматизована функція конвертації валют та індивідуальна діаграма прогнозування. Нижче наведено інтерфейс сайту CryptoPredictions.com (рис. 1.1) та приклад графіку прогнозу (рис. 1.2).

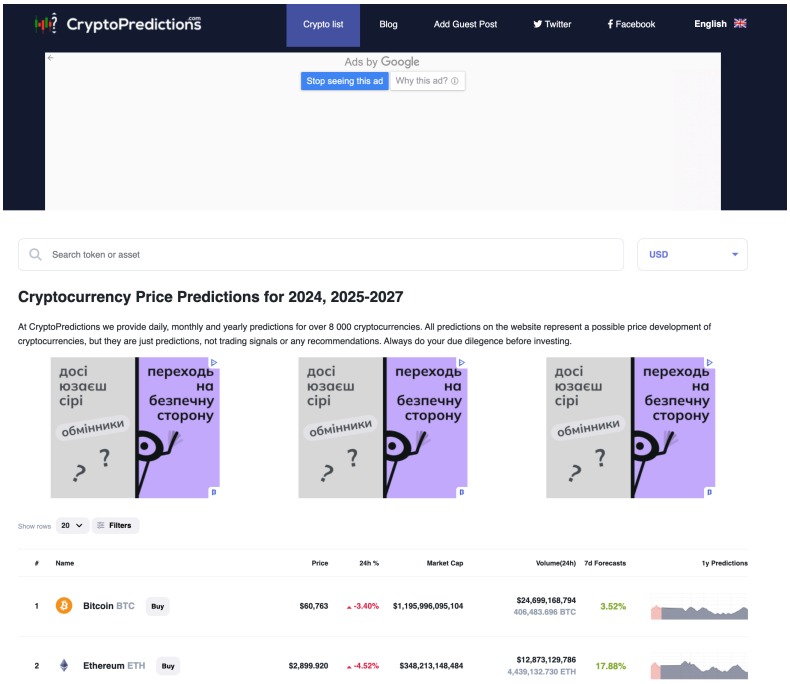

Рисунок – 1.1. Інтерфейс CryptoPredictions.com

Рисунок 1.2. – Графік прогнозу на CryptoPredictions.com

«WalletInvestor» – це інструмент для прогнозування цін на криптовалюту, що використовує алгоритми машинного навчання для надання щоденних та довгострокових прогнозів для понад 880 цифрових валют. Прогнози веб-сайту, що розгортаються протягом п'яти років, ґрунтуються на технічному аналізі, історичних ринкових даних та аналізі тенденцій.

Користувачі можуть легко переміщатися по сайту, вибираючи бажаний актив чи категорію. Наприклад, вибір розділу криптовалют призводить до виведення детальних графіків, майбутніх прогнозів та порівнянь цін.

Сайт надає щомісячні прогнози щоденних мінімумів, максимумів та середніх цін на криптовалюту. Однак прогнози для валют з меншими обсягами торгів зазвичай обмежені на наступні місяці. Титульну сторінку сайту ресурсу зображено на рис. 1.3.

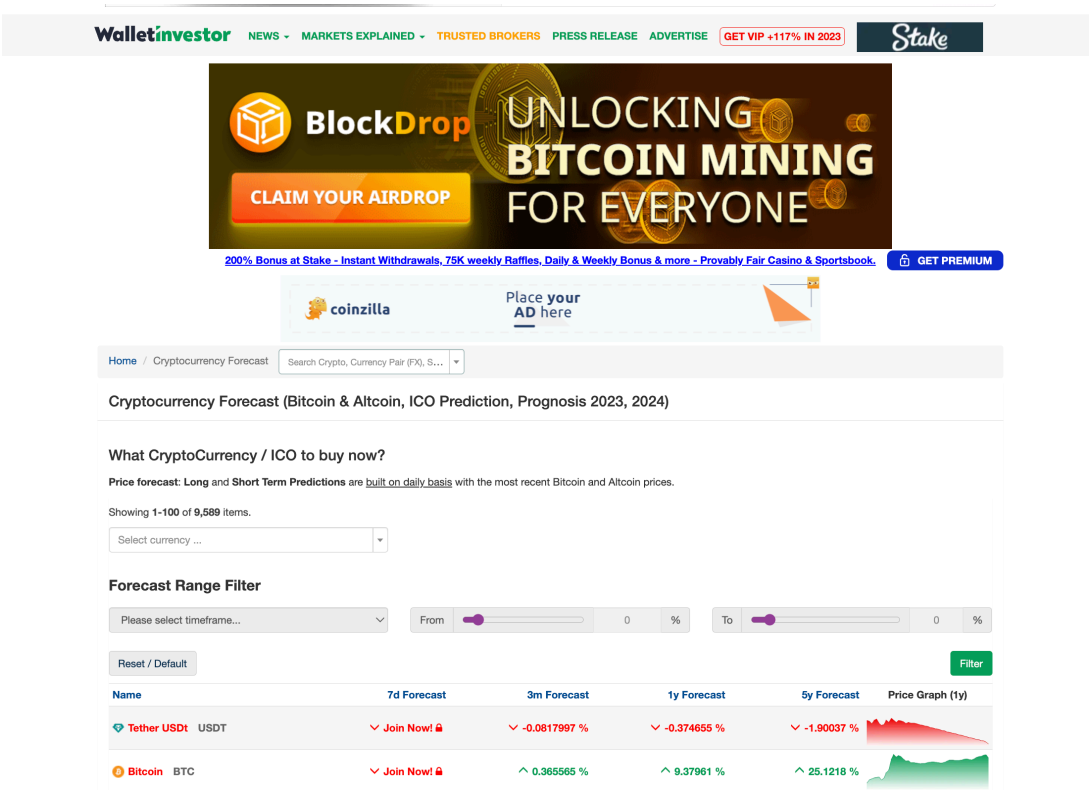

Рисунок 1.3. – Інтерфейс «Walletinvestor»

«NeuroShell» – це програмна для прогнозування змін фінансового ринку із застосуванням ШІ. Було враховано створення зручного інтерфейсу для роботи з нейронними мережами, уникнувши програмування.

Особливістю «NeuroShell» є оптимізація генетичними алгоритмами для визначення найкращих параметрів індикаторів та аналізу входів нейронних мереж. Основні етапи побудови нейронних мереж лишаються засекреченими.

Головна архітектура, яка використовується в, є багатошаровий персептрон. Інтерфейс «NeuroShell» зображено на рис. 1.4.

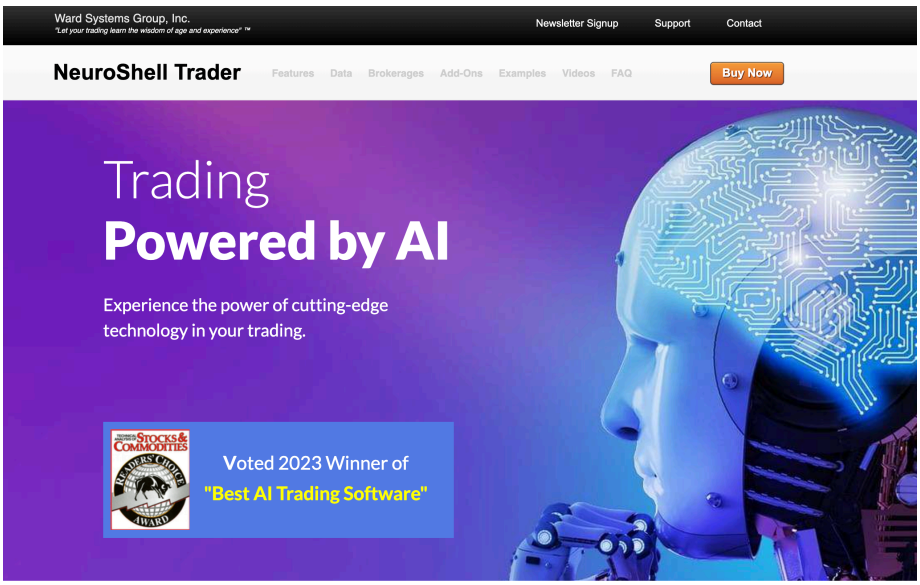

Рисунок 1.4. – Інтерфейс «NeuroShell»

«TradingBeasts» – це інструмент прогнозування криптовалют, який пропонує щоденні прогнози цін для різних криптовалют. Забезпечуючи як короткострокові, так і довгострокові стратегії інвестування, сайт оснащений рядом інструментів технічного аналізу, щоб допомогти криптовалютним інвесторам приймати обґрунтовані рішення.

«TradingBeasts» надає комплексні щомісячні прогнози, включаючи високі, низькі, середні та закриті ціни, а також відсоткові зміни для кожної криптовалюти.

Його прогнози ґрунтуються на даних реального ринку, зібраних з різних бірж, використовуючи алгоритмічні методи разом з лінійним та поліноміальним регресійними аналізами для формулювання прогнозів. Інтерфейс цього ресурсу можна розглянути на рис. 1.5.

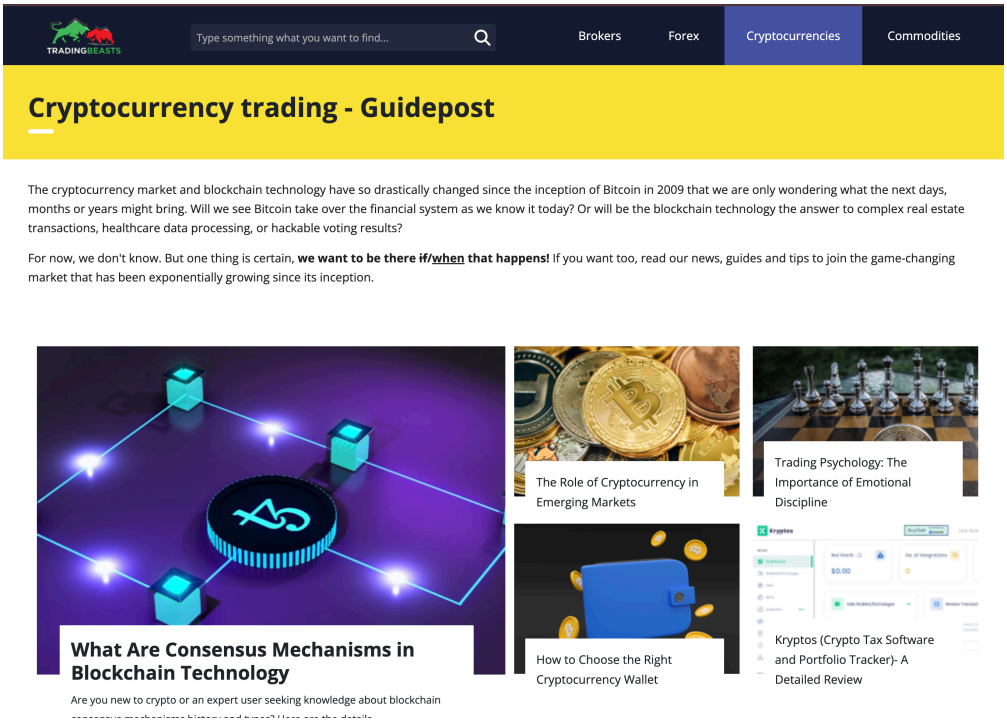

Рисунок 1.5. – Інтерфейс «TradingBeasts»

«Fetch.ai» – це платформа, яка поєднує ШІ і машинне навчання з технологією блокчейну. Основна мета «Fetch.ai» – оптимізувати бізнес-операції, такі як обробка даних та торгівля (трейдинг), шляхом автоматизації.

Платформа підтримує розробку самостійних агентів, які працюють як інструменти на основі ШІ. Fetch.ai призначений для управління різноманітними складними завданнями, такими як високорівнева аналітика, процеси прийняття рішень та прогнозування моделями. У межах своєї екосистеми різні боти та інструменти програмуються для взаємодії один з одним. Для транзакцій у цій системі використовується FET, її власна цифрова валюта. Інтерфейс на рис. 1.6.

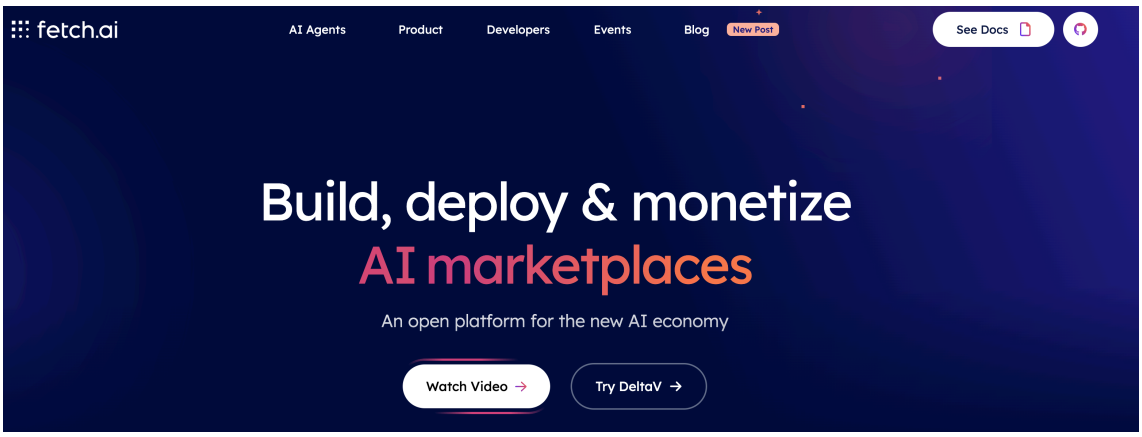

Рисунок 1.6. – Інтерфейс «Fetch.ai»

«Elliott Wave Forecast» – це програмний продукт для технічного аналізу фінансового ринку, який спеціалізується на застосуванні теорії хвиль Елліотта. Ця теорія широко застосовується для аналізу змін ціни на ринках, зокрема на ринку акцій, криптовалютах та інших.

Програмне забезпечення «Elliott Wave Forecast» допомагає трейдерам та інвесторам ідентифікувати хвилі Елліотта в ціновому графіку, застосовуючи спеціальні алгоритми та інструменти аналізу. Це програмне забезпечення може прогнозувати зміни щодо майбутніх цінових рухів на основі тенденцій, виявлених за допомогою теорії хвиль Елліотта.

Програмне забезпечення також надає додаткові функції та інструменти для технічного аналізу, такі як різноманітні індикатори, графіки, інтерактивні інтерфейси та інші. Вони є значною перевагою та допоміжними інструментами трейдерам у прийнятті рішень на ринку. Зображення інтерфейсу цього програмного продукту на рис. 1.7.

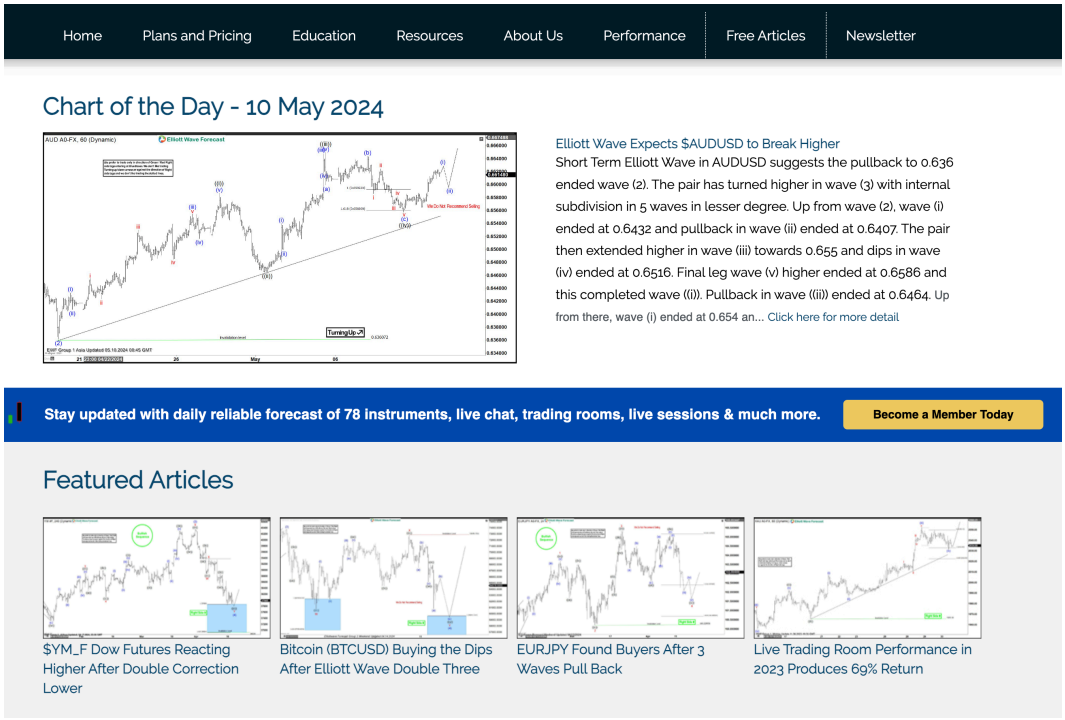

Рисунок 1.7. – Інтерфейс «Elliott Wave Forecast»

Це не кінцевий список сервісів, які надають можливість прогнозувати курс криптовалют. Збільшення кількості сервісів може свідчити про те, що ринок криптовалют розширюється та розвивається.

Висновки до розділу 1

У даному розділі було проведено дослідження предметної області. Розглянуто актіальність проблеми та її вирішення, та й загалом потенціал розширення ринку криптовалют та послуг, пов’язаних з ними. Було згадано короткі факти з історії створення, розвитку та поширення криптовалют, особливо біткоїна. Також наведено переваги фінансових операцій із криптовалютами і їхній вплив на міжнародну економіку.Було перелічено реалізованих програмних продуктів для прогнозування курсу криптовалют, які є в активному використанні багатьма юзерами.

РОЗДІЛ 2 ТЕОРЕТИЧНІ ДАНІ ДЛЯ ПОБУДОВИ ПРОЦЕСУ ПРОГНОЗУВАННЯ КУРСУ БІТКОЇНА

2.1 Методи, підходи та моделі для розв’язання задачі прогнозу

2.1.1 Штучний інтелект

Існує не одне визначення «штучного інтелекту» (англ. Artificial Intelligence, AI). Наукова спільнота досі не може дійти одного єдиного визначення, що є нормальним явищем. Було виділено два визначення, які, на думку автора, найбільш пасують.

ШІ – це технологія, що дозволяє комп'ютерам та машинам імітувати інтелект людини та розв'язувати задачі різного типу. Ідеальною характеристикою ШІ є його здатність раціоналізувати та вживати певні послідовності дій задля досягнення певної конкретної мети.

ШІ надає можливість машинам вчитися на досвіді, адаптуватися до нових вхідних даних і виконувати завдання, які зазвичай призначені людині. Більшість прикладів ШІ, про які можна почути сьогодні: від комп'ютерів, які грають у шахи, до автопілотів в автомобілях – зазвичай ґрунтуються на використанні глибокого навчання та обробки природної мови. З використанням цих технологій комп'ютери навчаються виконувати конкретні завдання, обробляючи значні обсяги даних та розпізнавати в них закономірності.

Трохи інше визначення було надано експертною групою Європейської комісії зі штучного інтелекту. ШІ – це розроблені людьми системи, які, отримавши комплексну мету, працюють у фізичному або цифровому світі, сприймаючи навколишнє середовище, інтерпретуючи зібрані дані різного типу. На основі знань,

які були отримані з цих даних, приймають найкращі рішення для досягнення заданої мети (зважаючи на попередньо визначені параметри) [8].

2.1.2 Машинне навчання

З еволюцією людства активно використовуються багато типів інструментів для виконання різних завдань простішим способом. Творчість людського мозку призвела до винаходу різних машин. Ці машини спростили життя людей, дозволяючи їм відповідати на різноманітні потреби.

Згідно з тим, яке означення дає Артур Самюель, машинне навчання (англ. Machine Learning, ML) – галузь досліджень, яка надає комп'ютерам здатність вчитися без явного програмування. Машинне навчання використовується для того, щоб навчитися ефективніше обробляти дані. Іноді після аналізу даних складно інтерпретувати отриману інформацію. Машинне навчання користується все більшим попитом, знаходиться все більше задач для його застосування. Було проведено багато досліджень щодо того, як навчити машини самостійно вчитися без явного програмування. Багато математиків і програмістів застосовують кілька підходів для знаходження рішення цієї проблеми [9].

Машинне навчання ґрунтується на різних алгоритмах для вирішення проблем з даними. Вчені-дослідники зауважують, що не існує жодного універсального алгоритму, який би ідеально підходив для вирішення будь-якої проблеми. Вибір алгоритму залежить від характеру проблеми, яку хоче вирішити людина, кількості змінних, типу моделі, яка найкраще підходить для неї, тощо.

2.1.3 Штучні нейронні мережі

Штучна нейронна мережа (англ. Artificial Neural Network, ANN) – це сукупність об’єднаних штучних вузлів, що вираховують та зберігають свій внутрішній стан. Цей вид обробки інформації був вигаданий на основі того, як біологічні нейронні системи обробляють її. Вузли мають зв'язки схожі на ті, що наявні у мозку людини між нейронами.

Штучний нейрон, отримуючи сигнал, обробляє його і може передавати цей сигнал іншим нейронам, якщо з ними наявні з’єднання. З’єднання називають ребрами (англ. edges). Нейрони та ребра зазвичай мають ваги (англ. weight), які пристосовуються в процесі навчання для корегування результатів. Вага збільшує або зменшує силу сигналу на з'єднанні. Допускається використання певних функції, щоб сигнал надсилався нейроном тільки у тому випадку, якщо було «перейдено» певний поріг значень.

Нейрони зазвичай об’єднані у шари, які виконують різні перетворення вхідних даних. Сигнал посліовно передається від шару до шару (від першого до останнього, інколи декілька разів).

У сфері ШІ штучні нейронні мережі успішно застосовуються для різного класу задач, наприклад, таких як: розпізнавання мови, аналізу зображень та відео, для створення програмних агентів або автономних роботів [10]. Нище наведено базову спрощену схему нейронної мережі (рис. 2.1).

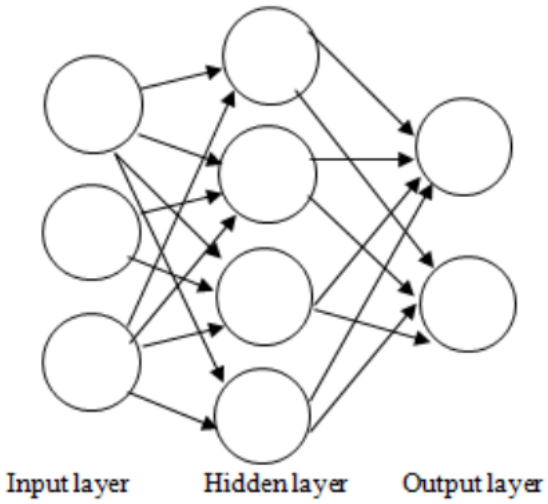

Рисунок 2.1. – Спрощена схема примітивної нейронної мережі

2.1.4 Глибинне навчання

Глибинне навчання – це окремий клас машинного навчання з використанням штучних нейронних мереж. Особливістю є використання декількох шарів для виділення ознак вищих рівнів із вхідних даних. Різні архітектури глибинного навчання, такі як глибинні нейронні мережі, глибинне навчання з підкріпленням, рекурентні нейронні мережі тощо, застосовується в таких областях, як обробка та розпізнавання природних мов, машинний переклад, класифікація семантичних висловлювань, розпізнавання рукописного письма, обробка аудіо та іншої мультимедійної інформації, інформаційного пошуку тощо. Моделі глибинного навчання набули популярностей у ріхних сферах. Вони мають перевагу при аналізі даних великого обсягу. Концепція глибинного навчання грунтується на застосуванні необмеженої кількості шарів обмеженого розміру, що надає можливість оптимізованої реалізації, при цьому зберігаючи універсальність.

Глибинне навчання застосовується багатьма відомими компаніями у своїх програмних продуктах, які обробляють величезні обсяги інформації. Наприклад, такі як «Google», «Facebook», «Apple». Вони ефективно реалізують та просувають

різні проєкти, пов'язані з глибинним навчанням. Наприклад, віртуальний особистий асистент «Siri» від «Apple», який надає широкий спектр послуг, включаючи пошук різної інформації за людським запитом, відповіді на запитання користувача та різні інші дрібні операції, як дзвінки, встановлення таймеру, нагадування тощо. А от «Google» використовує алгоритми глибокого навчання ддя обробки даних, отриманих з інтернету, що є неструктурованими та величезними за своїми обсягами, для перекладача «Google Translate», наприклад [11].

2.1.5 Рекурентні нейронні мережі

Рекурентна нейронна мережа (англ. Recurrent Neural Networks, RNN) – це тип штучних нейронних мереж, які відрізняються напрямком потоку інформації між шарами. Це двонаправлена штучна нейронна мережа. Це значить, що вона дозволяє вихідним даним з деяких вузлів впливати на подальший вхід у ті самі вузли. Використовується внутрішній стан (пам'ять) для обробки послідовностей вхідних даних, що допомагає вирішувати такі завдання, як розпізнавання мови тощо.

Людський мозок працює так, що людина має послідовне мислення. Читаючи будь-який текст, людина здатна зрозуміти зміст та значення кожного слова, відштовхуючись від контексту. Традиційні нейронні мережі не володіють цією властивістю і це їхній головний недолік. Уявімо, наприклад, що постала задача класифікування події у фільмі. Незрозуміло, чи могла б зробити це звичайна мережа, не беручи до уваги інформацію про попередні події у фільмі.

Для вирішення розглянутої проблеми якраз були придумані рекурентні нейронні мережі, які містять зворотні зв’язки та можуть зберігати інформацію.

В більшості випадків використовуються модифіковані мережі, як, наприклад, довга короткострокова пам’ять [12].

2.1.6 Довга короткострокова пам’ять

Довга короткострокова пам’ять (англ. Long short-term memory, LSTM) – вид рекурентних нейронних мереж, у якому реалізовано можливість навчання довгострокових зв’язків.

Будь-яка рекурентна нейронна мережа має вигляд ланцюга модулів нейронної мережі, що повторюються. Звичайні рекурентні нейронній мережі мають просту будову і складаються, наприклад, з одного шару із функцією активації тангенс (tanh). Представлено схематичну будову на рис. 2.2.

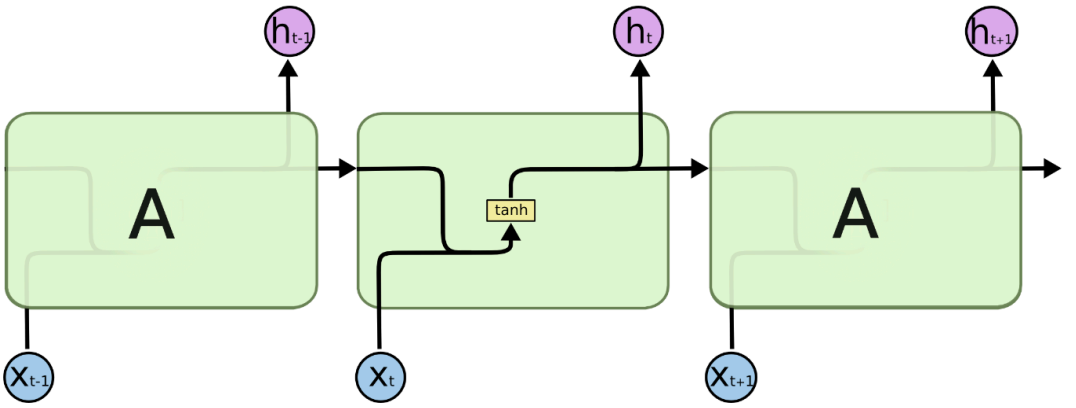

Рисунок 2.2. – Спрощена будова

На картинці зображено h_{t-1} та x_t як поточні вихід на вхід відповідно на кожному кроці.

Структура LSTM також є ланцюгом, але модулі мають більш складну структуру. Замість одного шару, як у звичайній рекурентній нейронній мережі, модуль LSTM містить чотири компоненти, які взаємодіють між собою (memory cell, forget gate, input gate, output gate). Будову LSTM представлено на рис. 2.3.

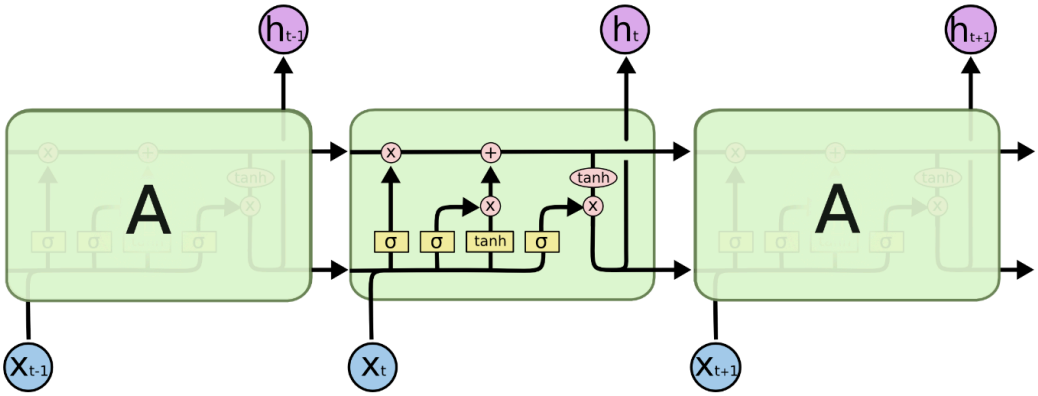

Рисунок 2.3. – Ускладнена будова

Комірка стану (memory cell) це верхня горизонтальна лінія на рисунку вище, інформація, яка там зберігається, регулюється так званими фільтрами (gates). Комірка стану проносить та зберігає усю інформацію протягом кожного кроку.

Спочатку відбувається відсіювання інформації з memory cell. За це відповідає forget gate layer. Цей фільтр зважає на прихований (попередній) стан h_{t-1} й вхідні дані x_t та передає їх до сигмоїдної функції. Ця функція вертає значення 0 або 1, що означають «повністю викинути» і «повністю пропустити» відповідно.

Наступний крок включає дві чатини. Спочатку input layer gate визначає, які значення треба оновити. Потім tanh-шар складає вектор потенційних значень-кандидатів $\tilde{C}_t$.

І наостанок використовуємо output gate layer, що завдяки сигмоїдному шару вирішує, які частини комірки стану додати до результату, і tanh-шар, домножений на результат із попередньої функції.

LSTM вміє запам’ятовувати інформацію на певний значний час, тому така модель є ефективною в прогнозуванні показників, які мають тенденцію до швидких змін. LSTM є ресурсозатратною через потребу зберігання даних та їхньої обробки, тому така модель потребує значних обчислювальних потужностей.

2.1.7 Трансформери

Трансформер – це архітектура глибокого навчання, розроблена компанією «Google» на основі механізму уваги типу multi-head, запропонованого у роботі 2017 року «Attention is All you Need» [13]. Текст перетворюється на числові дані, що називаються токенами, і кожен токен перетворюється у вектор за допомогою пошуку в таблиці відповідності слів. У кожному шарі токен контекстуалізується в межах вікна контексту з незамаскованими токенами за допомогою паралельного механізму уваги типу multi-head, що дозволяє збільшити вплив сигналу для ключових токенів та зменшити для менш важливих (тобто виділити важливе і відкинути неважливе) [14].

Трансформери мають перевагу відсутності рекурентних одиниць, і тому вимагають менше часу навчання, ніж попередні рекурентні нейронні архітектури, такі як LSTM, і їхні пізніші варіації широко застосовані для навчання великих мовних моделей на великих мовних наборах даних [15].

Ця архітектура наразі використовується в розв’язанні широкого спектру задач, таких, як обробка природних мов, обробці аудіо тощо. Також вона передувала розробці попередньо навчених систем, таких як породжувальний попередньо натренований трансформер (англ. Generative pre-trained transformer, GPT) та двоспрямоване кодувальне представлення із трансформерів.

2.1.8 Двоспрямоване кодувальне представлення із трансформерів

Двоспрямоване кодувальне представлення із трансформерів (англ. Bidirectional Encoder Representations from Transformers, BERT) — це методика машинного навчання, що ґрунтується на трансформері, для попереднього тренування обробки природної мови, розроблена «Google». BERT було створено й опубліковано 2018 року Джейкобом Девліним та його колегами з «Google». З 2019

рік «Google» використовує BERT для кращого розуміння пошуку користувачів [16,17].

На відміну від попередніх моделей представлення мови, BERT призначений для попереднього навчання глибоких двонаправлених представлень з невідмічених текстів, враховуючи повний контекст справа і зліва на всіх шарах. В результаті попередньо навчена модель BERT може бути доналаштована з всього лише одним додатковим вихідним шаром, щоб створити модель для широкого спектру задач, таких як відповіді на запитання тощо, без суттєвих модифікацій архітектури, специфічних для задачі [16,17].

Оригінальна англомовна модель BERT існує у двох наперед натренованих варіантах:

1) базова (BASE) модель BERT, нейромережна архітектура з 12 шарами, 768 прихованими, 12 головами, 110 мільйонами параметрів;

2) велика (LARGE) модель BERT, нейромережна архітектура з 24 шарами, 1024 прихованими, 16 головами, 340 мільйонами параметрів.

Обидві було натреновано на BooksCorpus (датасет, набір даних з різних видань з інтернету) з 800 мільйонами слів, та одній з версій англійської Вікіпедії з 2 500 мільйонами слів. Нижче наведено архітектуру моделі (рис. 2.4).

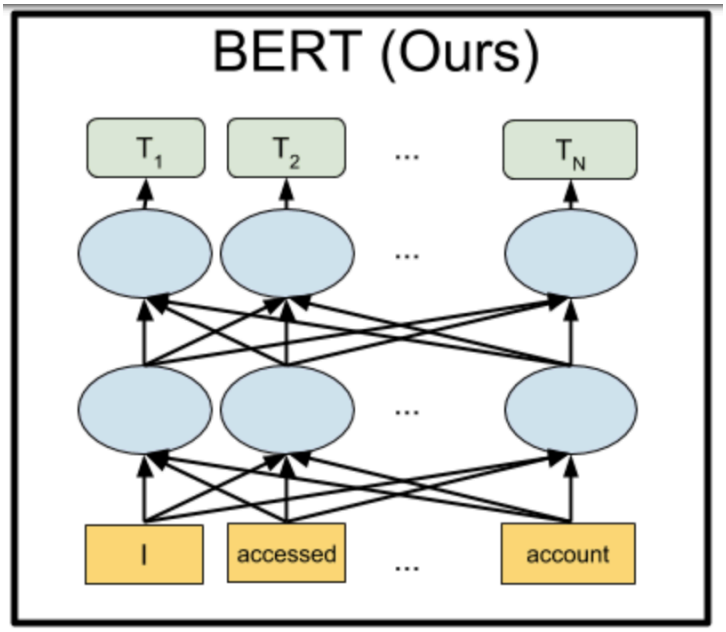

Рисунок 2.4. – Архітектура

Для використання попередньо навчених мовних моделей існують два підходи.

1. У підході на основі ознак (англ. feature-based) представлення попередньо навченої моделі використовуються як додаткові функції для архітектури іншої моделі.
2. У підході з точним налаштуванням (англ. fine-tuning) мовні представлення використовуються для конкретних задач після точного налаштування всіх попередньо навчених параметрів моделі.

Використання BERT містить два етапи (наведено на рис. 2.5).

1. Попереднє навчання – модель навчається на невідмічених даних, виконуючи різні завдання.
2. Точне налаштування – модель завантажується з попередньо навченими параметрами і навчається на відмічених даних.

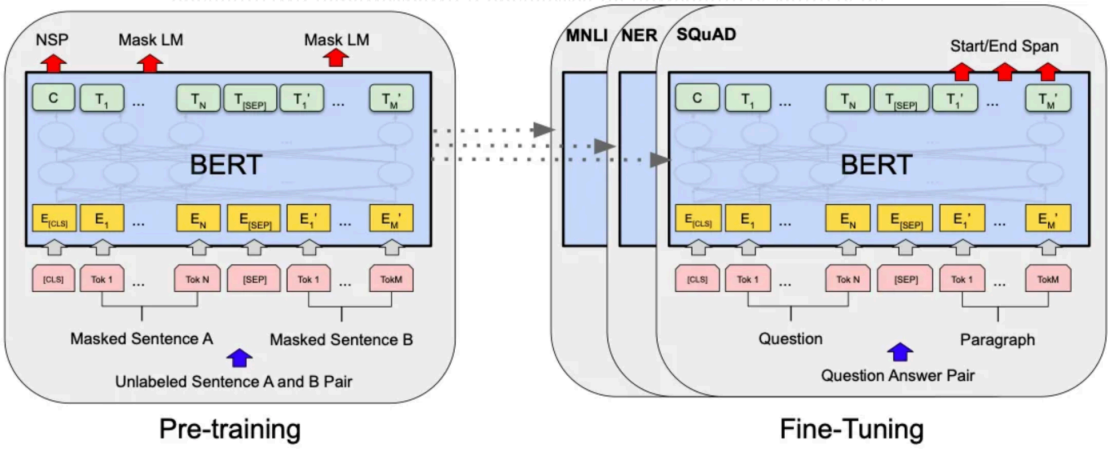

Рисунок 2.5. – Етапи BERT

Для розв’язання даної задачі було вирішено застосувати LSTM та BERT для глибокого машинного навчання. Вхідні дані, які будуть описані нижче, містять як природну мову, так і числовий формат даних. Застосування комбінації таких підходів може, на думку автора, гарно впоратися із поставленою задачею прогнозування.

2.2 Підходи до обробки та підготовки даних

Зазвичай використовується 2 основних підходи до перетворення числових даних в «однорідний» вигляд: нормалізація та стандартизація.

Стандартизація – це перетворення ознак шляхом віднімання середнього значення і ділення на стандартне відхилення. Формула:

$$x_{new} = \frac{x - x_{mean}}{x_{std}}, \tag{2.1}$$

де x – значення до стандартизації;

x_{mean} – середнє значення;

x_{std} – стандартне відхилення.

Нормалізація – це підгонка ознак в діапазон $[0, 1]$, шляхом віднімання мінімального значення вибірки, а потім ділення результату на різницю максимального та мінімального. Формула:

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}, \tag{2.2}$$

де x – значення до нормалізації;

x_{min} – мінімальне значення;

x_{max} – максимальне значення.

Основна різниця між вказаними методами наведена нижче.

1. На нормалізовані дані на відміну від стандартизованих сильно впливають викиди.
2. Діапазон нормалізованих даних визначений заздалегіть, а от стандартизовані дані не обмежуються чітким діапазоном.
3. Нормалізація ефективна, коли нічого не знаємо про розподіл даних, а стандартизація – для нормального розподілу.

2.3 Метрики оцінки якості отриманих результатів

MSE (Mean Squared Error) – середнє квадратичне відхилення. Обраховується як сума квадратів відхилень, поділених на кількість спостережень.

Якщо прогноз даних достатньо точний, то ця оцінка якомога ближча до нуля.

Формула:

$$MSE = \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}, \tag{2.3}$$

де y_i – дійсні значення;

$\hat{y}_i$ – прогнозовані значення;

n – кількість спостережень.

RMSE (Root Mean Square Error) – корінь середнього квадратичного відхилення. Обчислюється як MSE під коренем. Формула:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}, \tag{2.4}$$

де y_i – дійсні значення;

$\hat{y}_i$ – прогнозовані значення;

n – кількість спостережень.

MAE (Mean Absolute Deviation) – середнє абсолютне відхилення. Обраховується як сума модулів різниць між прогнозованим і дійсним значеннями, які поділені на дійсне значення, уся сума ділиться на кількість спостережень.

Якщо прогноз даних достатньо точний, то ця оцінка якомога ближча до нуля.

Формула:

$$MAE = \frac{\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right|}{n}, \tag{2.5}$$

де y_i – дійсні значення;

$\hat{y}_i$ – прогнозовані значення;

n – кількість спостережень.

Висновки до розділу 2

У цьому розділі було описано теоретичну частину для розв’язання задачі прогнозу криптовалюти на основі новин. Було наведено моделі та методики, які є актуальним та широко використовуються як підходід розв’язання поставленої задачі. Наведено оцінки якості отриманих результатів, за якими визначатимуться достовірність прогнозованих значень.

Основними технологіями, що будуть застосовані в нейронній мережі є BERT та LSTM. Було розглянуто основну ідею їхньої роботи та функціонал, який вони можуть надати в контексті поставленої задачі.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Опис процедури побудови нейронної мережі

Першим етапом є обробка тексту за допомогою BERT. Побудована нейронна мережа приймає вхідні текстові дані (попередньо перетворені в список ідентифікаторів токенів та список індексів уваги) через модель BERT. Результатом є вектор ембедингів BERT фіксованої довжини, який містить узагальнену інформацію про текст.

Отримані дані проходять через First Fully Connected шар (fc1), де вони трансформуються за допомогою лінійного перетворення, а потім через шар нормалізації (bn1, batch normalization) та активації relu, який перетворює вхідний сигнал на 0, якщо значення від’ємне, а інакше не змінює значення.

Далі дані часових рядів проходять через два рекурентних LSTM шари (lstm1 та lstm2), які дозволяють враховувати залежності в часових рядах та відобразити їх у вектори ознак. Результатом є тензор, що містить закодовану інформацію про залежності.

Наступним кроком є об'єднання ознак. Функція torch.max вибирає максимальне значення з кожного стовпчика в матриці, отриманій з рекурентного LSTM шару. Результатом є тензор global_max_pooling, в якому кожен рядок містить максимальне значення у відповідному стовпчику x_time_series. torch.mean обчислює середнє значення для кожного стовпчика в матриці, отриманій з рекурентного LSTM шару. Результатом є тензор global_avg_pooling, в якому кожен рядок містить середнє значення у відповідному стовпчику x_time_series.

Обидва отримані результати global_max_pooling та global_avg_pooling разом з результатом обробки тексту через BERT об'єднуються в один тензор. Це робиться за допомогою функції torch.cat, яка зливає тензори.

Об'єднаний вектор ознак проходить через додаткові Fully Connected шари (fc2, fc3), де він проходить лінійне перетворення. Спочатку другий шар, потім нормалізація, аналогічна до попередньої, а далі шар активації relu. Додається dropout шар, щоб уникати перенавчання. Цей механізм випадковим чином обнуляє елементи незалежно для кожного виклику. Далі третій шар і шар активації relu.

Остаточний вихід після обробки усіма шарами подається на два шари (fc_min, fc_max), які здійснюють прогнози для мінімальних та максимальних значень ціни відповідно.

На рис. 3.1 зображено основну архітектуру побудованої нейронної мережі, виведено за допомогою функції modules() пакету pytorch.

```
(bert): BertModel(
  (embeddings): BertEmbeddings(
    (word_embeddings): Embedding(28996, 768, padding_idx=0)
    (position_embeddings): Embedding(512, 768)
    (token_type_embeddings): Embedding(2, 768)
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (encoder): BertEncoder(
    (layer): ModuleList(
      (0-11): 12 x BertLayer(
        (attention): BertAttention(
          (self): BertSelfAttention(
            (query): Linear(in_features=768, out_features=768, bias=True)
            (key): Linear(in_features=768, out_features=768, bias=True)
            (value): Linear(in_features=768, out_features=768, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): BertSelfOutput(
            (dense): Linear(in_features=768, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): BertIntermediate(
          (dense): Linear(in_features=768, out_features=3072, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): BertOutput(
          (dense): Linear(in_features=3072, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
    )
  )
  (pooler): BertPooler(
    (dense): Linear(in_features=768, out_features=768, bias=True)
    (activation): Tanh()
  )
)
(fc1): Linear(in_features=768, out_features=64, bias=True)
(bn1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(relu): ReLU()
(dropout): Dropout(p=0.5, inplace=False)
(lstm1): LSTM(3, 128, batch_first=True, bidirectional=True)
(lstm2): LSTM(256, 128, batch_first=True, bidirectional=True)
(fc2): Linear(in_features=576, out_features=64, bias=True)
(bn2): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(fc3): Linear(in_features=64, out_features=64, bias=True)
(fc_min): Linear(in_features=64, out_features=1, bias=True)
(fc_max): Linear(in_features=64, out_features=1, bias=True)
```

Рисунок 3.1. – Архітектура нейронної мережі

3.2 Робота із вхідним датасетом

3.2.1 Опис вхідних даних

Вхідний датасет містить наступну важливу для нас інформацію про новини, тобто їхній зміст природною мовою, та курс біткоїна (фактичний поточний, мінімальні та максимальні значення за певний період часу у майбутньому).

Було проведено попередній аналіз даних (англ. Exploratory data analysis, EDA), мета якого є викриття загальних закономірностей, характеру та властивостей даних, які аналізуються.

Обсяг датасету – 5000 об’єктів.

Вхідний датасет містить 8 полів. Короткий опис полів вказано нижче:

1) «caption» – заголовок новини, текстове поле;
2) «short_content» – короткий зміст новини, текстове поле;
3) «tags» – мітки з позначенням криптовалюти, текстове поле;
4) «mark_price» – ціна біткоїна, відображення оцінки поточної ринкової ціни активу, числове поле;
5) «1hour_future_price_min» – мінімальна ціна біткоїна за 1 годину, числове поле;
6) «1hour_future_price_max» – максимальна ціна біткоїна за 1 годину, числове поле;
7) «12hour_future_price_min» – мінімальна ціна біткоїна за 12 годин, числове поле;
8) «12hour_future_price_max» – максимальна ціна біткоїна за 12 годин, числове поле.

3.2.2 Очистка та підготовка вхідних даних

Підготовка даних передбачала виявлення пропусків та модифікація полів під нашу задачу. Виявилось, що ознака «tags» має 1169 пропусків. Оскільки для нас важливим є маркування поля «tags» для криптовалюти біткоїн, то було вирішено перетворити дані з текстового формату до числового із двома можливими значеннями: «0» – немає тегу «$BTC», «1» – наявний тег «$BTC», «2» – відсутнє значення.

Далі наведемо розподіли деяких ознак («mark_price» на рис. 3.2 та «tags» на рис. 3.3).

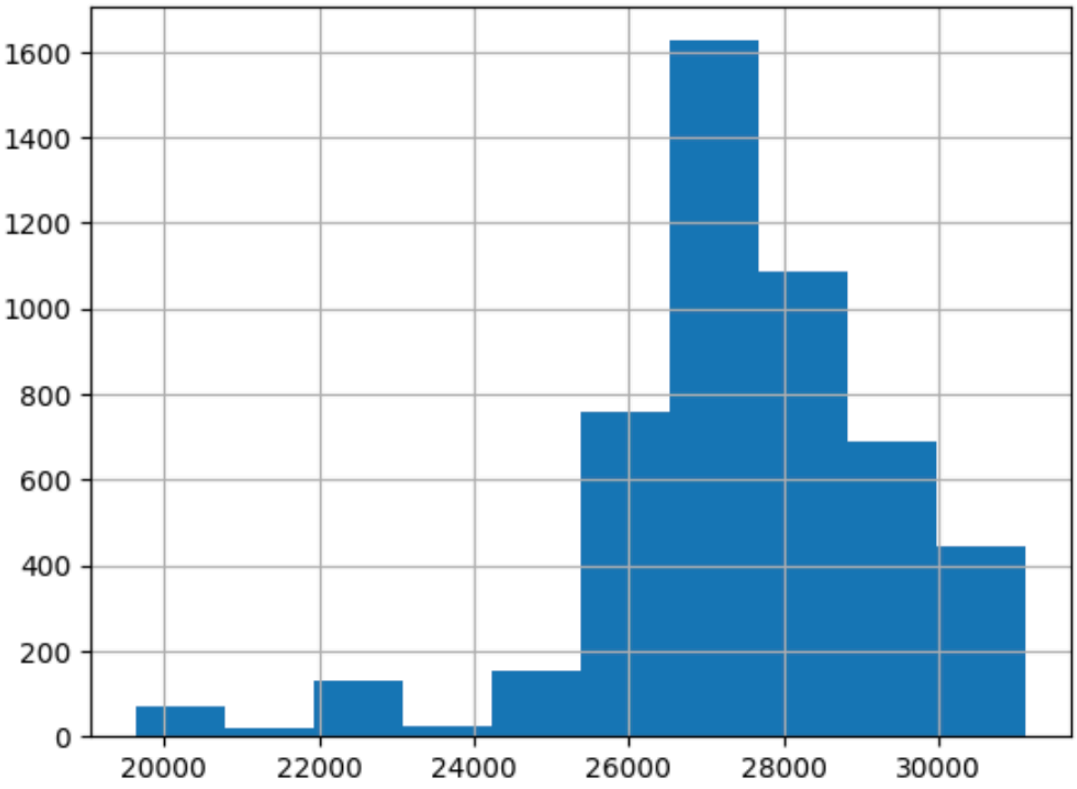

Рисунок 3.2. – Розподіл «mark_price»

Спостерігається схожість на нормальний розподіл, більша частина значень знаходиться в діапазоні від 26000 до 30000.

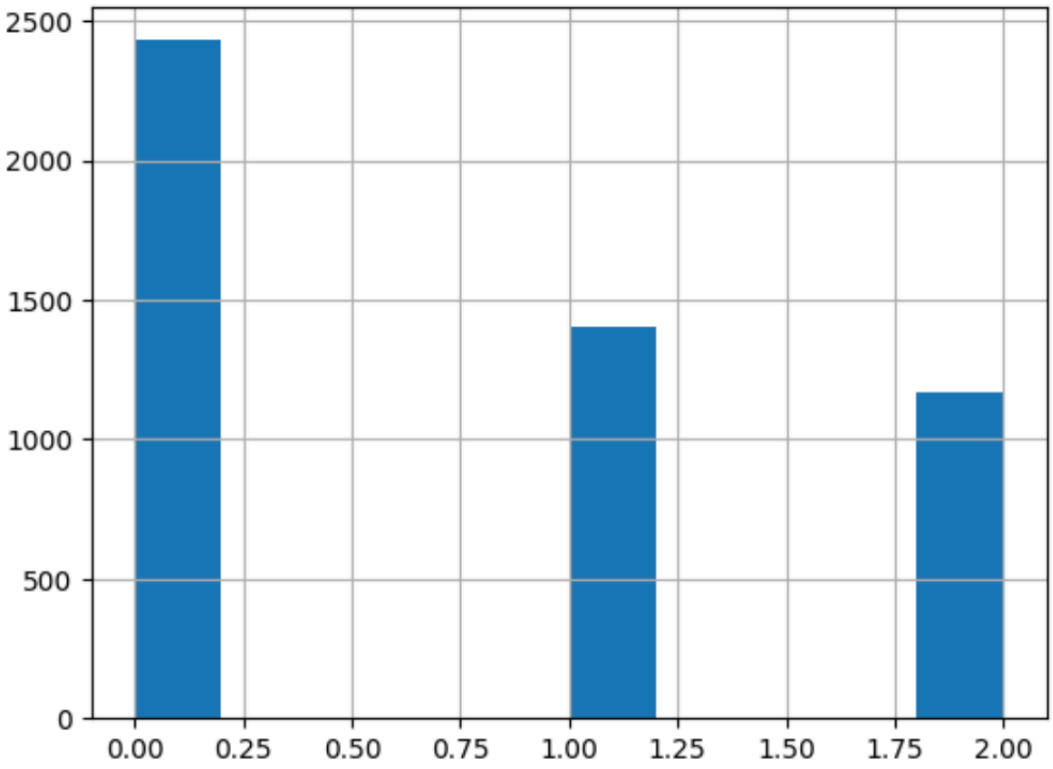

Рисунок 3.3. – Розподіл «tags»

За даним графіком видно, що половина значень поля «tags» має мітку «$BTC», приблизно чверть – не має, інша чверть – не має інформації про це.

Продемонструю перших 100 екземплярів і значення їхніх 3 ознак («mark_price», «1hour_future_price_min», «1hour_future_price_max») на графіку нижче (рис. 3.4).

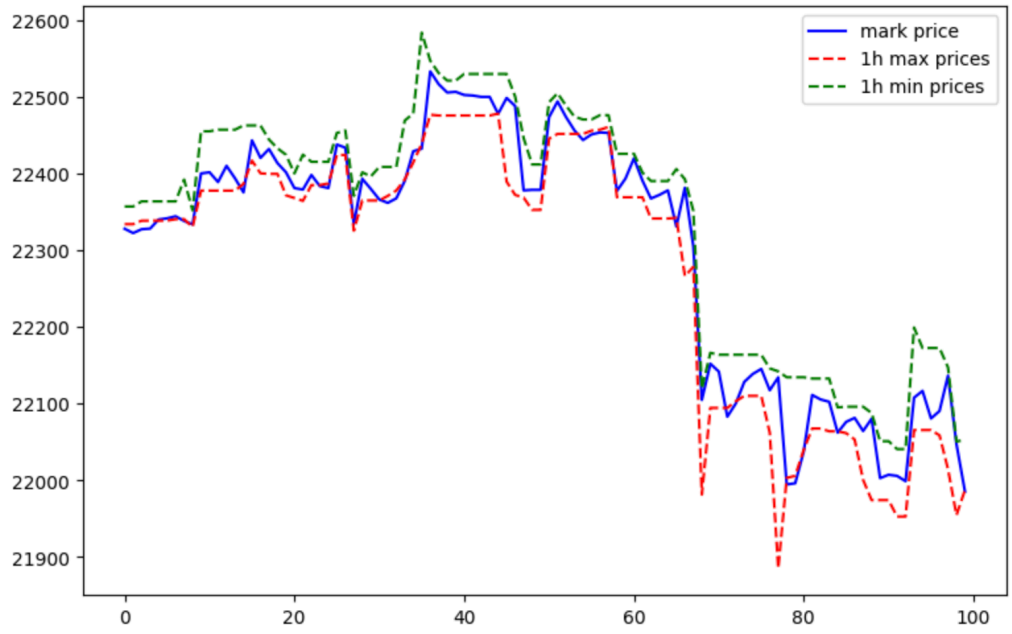

Рисунок 3.4. – Графік ознак «mark_price», «1hour_future_price_min», «1hour_future_price_max»

Аналогічно 3 ознаки, для інших полів прогнозу («mark_price», «12hour_future_price_min», «12hour_future_price_max») на графіку нижче (рис. 3.5).

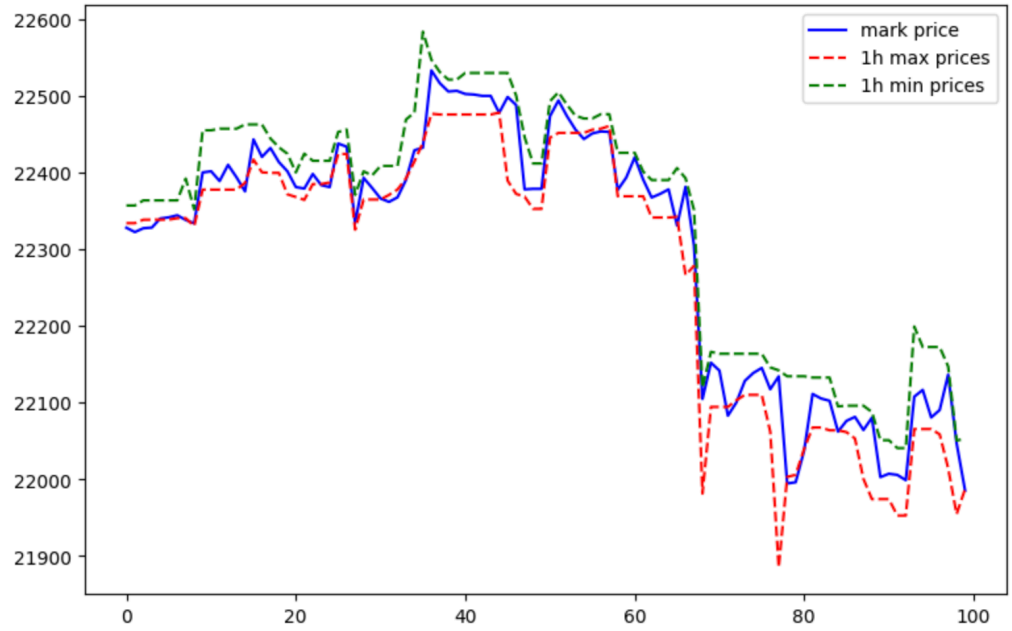

Рисунок 3.5. – Графік ознак «mark_price», «12hour_future_price_min», «12hour_future_price_max»

Також значення «mark_price» на всьому обсязі даних на рис. 3.6.

Рисунок 3.6. – Графік «mark_price» на всьому датасеті

Дані (текстові поля з природною мовою), які подаватимуться в модель BERT, були конкатиновані, як текстове поле «content» у наступному вигляді: *caption: **[значення caption]**, content: **[значення short_content]***. Оскільки вхід має обмеження, то на перше місце було подано важливішу, на мою думку, інформацію, а саме «caption», оскільки це поле містить стиснутий коротко викладений зміст новини.

Було стандартизовано числові дані.

Дані часового ряду були переформатовані так, що кожен екземпляр містить «вікно» даних у розмірі останніх 10 одиниць.

Виділено 2 вектори цільових значень для двох процесів навчань: («1hour_future_price_min», «1hour_future_price_max») і («12hour_future_price_min», «12hour_future_price_max»).

Далі відбувається певний «енкодинг» текстових даних (токенізація «caption»). Дані обрізаються до максимального розміру, який приймається BERT моделлю (було обрано значення 256), або доповнюються «порожніми» PAD токенами задля досягнення однакової розмірності. Додається «[CLS]» токен на

початок, а «[SEP]» токен у кінець. Після перетворення даних функція-токенізатор вертає список ідентифікаторів токенів, які будуть передаватися в модель, і список індексів, які вказують, на які токени повинна звертати увагу модель.

Перед навчання було розбито дані без перемішування на тренувальні та валідаційні обсягом 80% та 20% відповідно.

Використаємо клас DataLoader із бібліотеки PyTorch для певного представлення даних партіями. DataLoader дозволяє завантажувати дані партіями, що дозволяє оптимізувати використання пам'яті та прискорити процес навчання, особливо при роботі з великими обсягами даних. Гарним доповненням є перемішування даних перед кожною епохою, щоб уникнути перенавчання моделі та покращити загальну ефективність.

3.2 Результати роботи

3.2.1 Прогнозування мінімальної та максимальної ціни за 1 годину

Спершу розглянемо результати виконання програми для прогнозу наступних значень: («1hour_future_price_min», «1hour_future_price_max»).

Емпіричним методом було визначено максимальну кількість епох навчання та валідації (30 епох), а також критерії попередньої зупинки за умови, що модуль різниці значень валідаційної та тренувальної оцінки буде меншим за 0.005 та значення тренувальної оцінки менше за 0.035.

Виведення прогресу роботи програми можемо побачити на рис. 3.7.

```
Epoch [1/30], Training Loss: 0.25827226595254615, RMSE: 0.5082049448328363
Epoch [1/30], Validation Loss: 0.017793720404151826, RMSE: 0.13339310478488695
Epoch [2/30], Training Loss: 0.157513278957922, RMSE: 0.39687942622151884
Epoch [2/30], Validation Loss: 0.07542449260829016, RMSE: 0.2746351991429543
Epoch [3/30], Training Loss: 0.1431942543736659, RMSE: 0.37841016684764944
Epoch [3/30], Validation Loss: 0.07072969415225089, RMSE: 0.26595054832101944
Epoch [4/30], Training Loss: 0.12578706751693972, RMSE: 0.3546647255041579 5
Epoch [4/30], Validation Loss: 0.03140099229407497, RMSE: 0.1772032513642878
Epoch [5/30], Training Loss: 0.11706738672102801, RMSE: 0.34215111679056087
Epoch [5/30], Validation Loss: 0.05068691518856212, RMSE: 0.22513754726513774
Epoch [6/30], Training Loss: 0.10150355593883433, RMSE: 0.3185962271258628
Epoch [6/30], Validation Loss: 0.10788440207950771, RMSE: 0.3284576107803071
Epoch [7/30], Training Loss: 0.08868949662777595, RMSE: 0.2978078182784595
Epoch [7/30], Validation Loss: 0.053533177293429615, RMSE: 0.2313723779828301
Epoch [8/30], Training Loss: 0.0839990250871051, RMSE: 0.2898258530343784
Epoch [8/30], Validation Loss: 0.062128388602286574, RMSE: 0.2492556691477379
Epoch [9/30], Training Loss: 0.06550799905322492, RMSE: 0.25594530480793143
Epoch [9/30], Validation Loss: 0.06038659478072077, RMSE: 0.24573684050365904
Epoch [10/30], Training Loss: 0.06258385805529543, RMSE: 0.25016765989091283
Epoch [10/30], Validation Loss: 0.016297508837014904, RMSE: 0.127661696828042
Epoch [11/30], Training Loss: 0.05331702221534215, RMSE: 0.23090479036897904
Epoch [11/30], Validation Loss: 0.08898386187152937, RMSE: 0.2983016290125305
Epoch [12/30], Training Loss: 0.04996361918398179, RMSE: 0.22352543296900643
Epoch [12/30], Validation Loss: 0.050103005974087865, RMSE: 0.22383700760617728
Epoch [13/30], Training Loss: 0.04206362341239583, RMSE: 0.20509418181020111
Epoch [13/30], Validation Loss: 0.04675022457784508, RMSE: 0.21621800243699663
Epoch [14/30], Training Loss: 0.04019098477845546, RMSE: 0.20047689337790392
Epoch [14/30], Validation Loss: 0.03725194641185226, RMSE: 0.1930076330403859
Epoch [15/30], Training Loss: 0.038502363183069974, RMSE: 0.19622019055915213
Epoch [15/30], Validation Loss: 0.023945502402202692, RMSE: 0.1547433436442508
Epoch [16/30], Training Loss: 0.036475639788550326, RMSE: 0.19098596751738156
Epoch [16/30], Validation Loss: 0.04911749061546288, RMSE: 0.2216246615687498
Epoch [17/30], Training Loss: 0.03323397051513893, RMSE: 0.1823018664609305
Epoch [17/30], Validation Loss: 0.017048592382634523, RMSE: 0.13057025841528583
Epoch [18/30], Training Loss: 0.032155437142355366, RMSE: 0.17931937191044187
Epoch [18/30], Validation Loss: 0.02538877986778971, RMSE: 0.159338569931419
Epoch [19/30], Training Loss: 0.03322113320522476, RMSE: 0.18226665412308626
Epoch [19/30], Validation Loss: 0.05414481651678216, RMSE: 0.23269038767594624
Epoch [20/30], Training Loss: 0.03862197715090588, RMSE: 0.19652474946150136
Epoch [20/30], Validation Loss: 0.017541867637773975, RMSE: 0.13244571581509904
Epoch [21/30], Training Loss: 0.03246473899227567, RMSE: 0.18017974079311933
Epoch [21/30], Validation Loss: 0.03281388620496727, RMSE: 0.18114603557618167
Finished Training
```

Рисунок 3.7. – Логи тренування та валідації для прогнозу значень за 1 годину

Спостерігається, що процес зупинився із ранньою зупинкою після 21-ої епохи. На наступних графіках виведемо прогнозовані та реальні значення змінних (синім – реальні значення, червоним – прогнозовані) на всьому тестовому наборі (рис. 3.8, рис. 3.9).

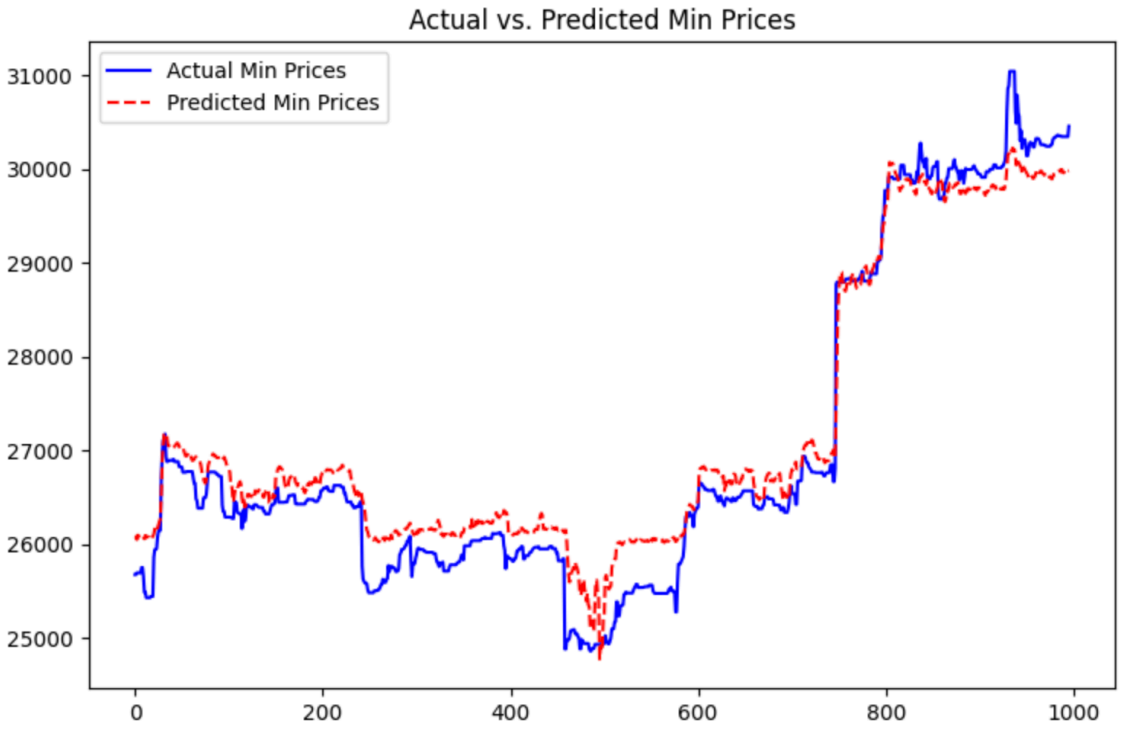

Рисунок 3.8. – Графік мінімальних реальних та прогнозованих значень

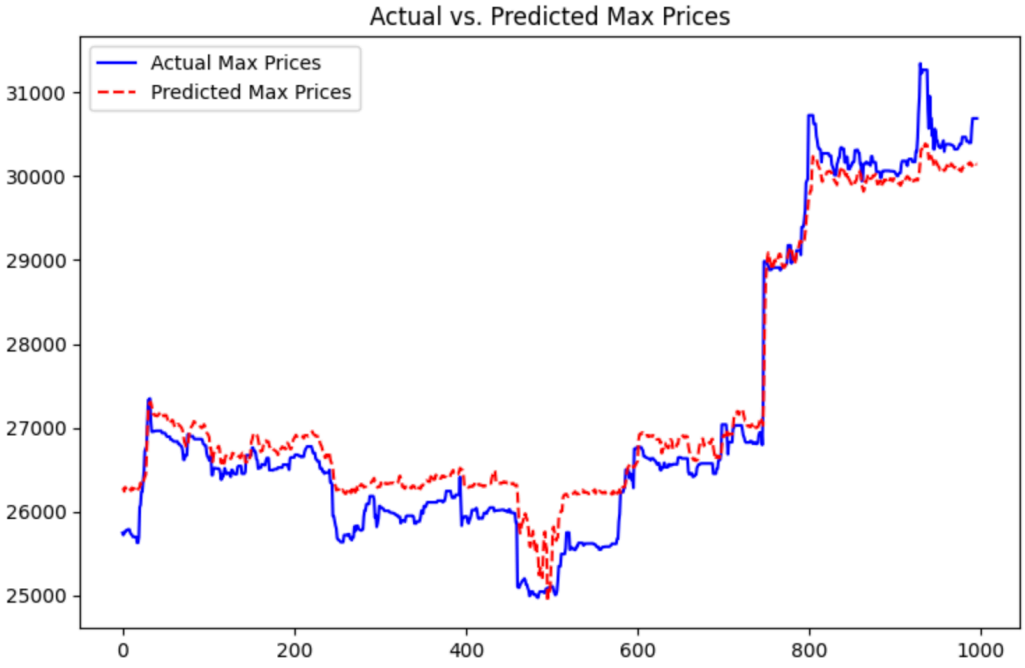

Рисунок 3.9. – Графік максимальних реальних та прогнозованих значень

Для більш детального огляду відмалюємо перших 200 значень (рис. 3.10, рис. 3.11).

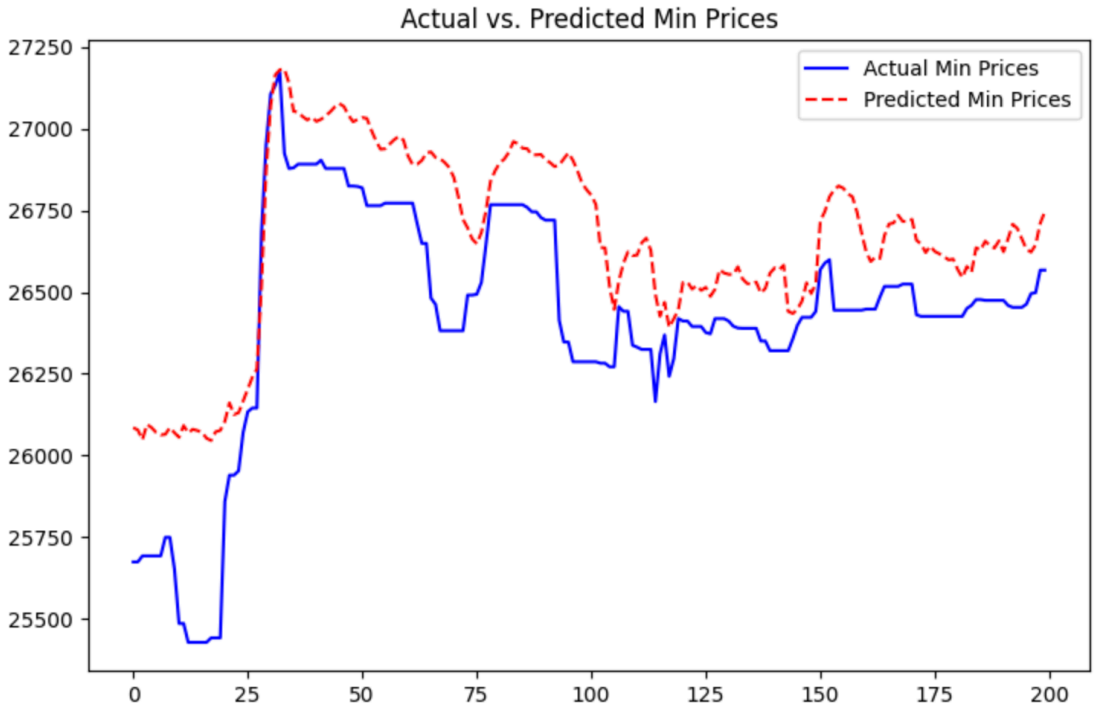

Рисунок 3.10. – Графік 200 мінімальних реальних та прогнозованих значень

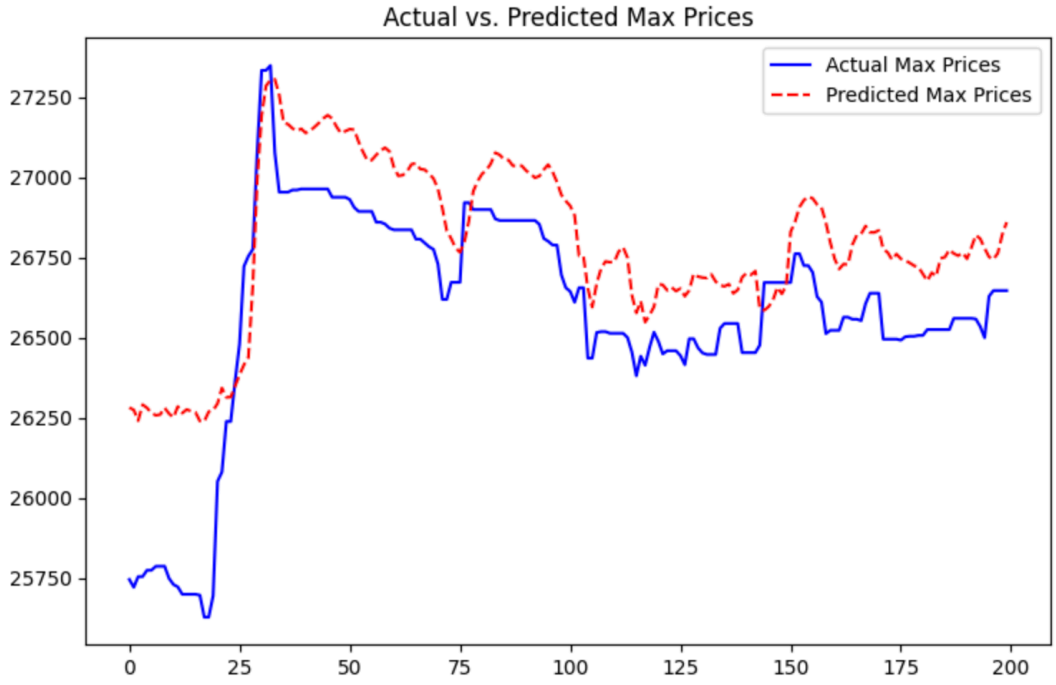

Рисунок 3.11. – Графік 200 максимальних реальних та прогнозованих значень

Оцінку якості моделі було проведено за допомогою наступних метрик якості, результати яких виведено в таблицю 3.1.

Таблиця 3.1. – Оцінки якості моделі для прогнозу за 1 годину

MSE	RMSE
0.033	0.181

Загалом можемо вважати дану модель прийнятною, а результати прогнозу достатньо точними. Тенденція зміни курсу гарно прогнозується, що є важливим у даній задачі, оскільки точні показники спрогнозувати складно, а от динаміка зростання або спадання доволі точна.

3.2.2 Прогнозування мінімальної та максимальної ціни за 12 годин

Тепер розглянемо результати виконання програми для прогнозу наступних значень: («12hour_future_price_min», «12hour_future_price_max»).

Використаємо попередні критерії зупинки.

Виведення прогресу роботи програми можемо побачити на рис. 3.12.

```
Epoch [1/25], Training Loss: 0.26493378384038807, RMSE: 0.5147171882115343
Epoch [1/25], Validation Loss: 0.1133495043698349, RMSE: 0.33667418132347915
Epoch [2/25], Training Loss: 0.1770106403948739, RMSE: 0.42072632481801503
Epoch [2/25], Validation Loss: 0.12721686675125965, RMSE: 0.3566747352298164
Epoch [3/25], Training Loss: 0.17946616729721426, RMSE: 0.42363447368836055
Epoch [3/25], Validation Loss: 0.19657294878503306, RMSE: 0.4433654799203847
Epoch [4/25], Training Loss: 0.15207046326715498, RMSE: 0.3899621305552053
Epoch [4/25], Validation Loss: 0.06301960397977382, RMSE: 0.2510370569851667
Epoch [5/25], Training Loss: 0.14268002609489486, RMSE: 0.37773009688783715
Epoch [5/25], Validation Loss: 0.16204003321006893, RMSE: 0.4025419645329775
Epoch [6/25], Training Loss: 0.13835048069711775, RMSE: 0.3719549444450467
Epoch [6/25], Validation Loss: 0.11611874414957128, RMSE: 0.34076200514372385
Epoch [7/25], Training Loss: 0.11817900630179792, RMSE: 0.3437717357517891
Epoch [7/25], Validation Loss: 0.07835080125019886, RMSE: 0.279912131302305
Epoch [8/25], Training Loss: 0.10519035413628444, RMSE: 0.32433062472773744
Epoch [8/25], Validation Loss: 0.09652750383014791, RMSE: 0.3106887571672781
Epoch [9/25], Training Loss: 0.09856223654933274, RMSE: 0.3139462319400135
Epoch [9/25], Validation Loss: 0.06795580180798425, RMSE: 0.2606833362683243
Epoch [10/25], Training Loss: 0.09704794241348282, RMSE: 0.3115251874463489
Epoch [10/25], Validation Loss: 0.059252273266902196, RMSE: 0.24341789841115258
Epoch [11/25], Training Loss: 0.09242798784514888, RMSE: 0.30401971621121693
Epoch [11/25], Validation Loss: 0.09493525572470389, RMSE: 0.30811565316404144
Epoch [12/25], Training Loss: 0.08590546317631378, RMSE: 0.293096337705393
Epoch [12/25], Validation Loss: 0.10128223293460906, RMSE: 0.31824869667385763
Epoch [13/25], Training Loss: 0.08511014832183719, RMSE: 0.29173643639737085
Epoch [13/25], Validation Loss: 0.15059800879680552, RMSE: 0.3880695927237865
Epoch [14/25], Training Loss: 0.0844935384276323, RMSE: 0.29067772262014213
Epoch [14/25], Validation Loss: 0.08660206948232371, RMSE: 0.2942822955638407
Epoch [15/25], Training Loss: 0.07260200077435001, RMSE: 0.26944758446560624
Epoch [15/25], Validation Loss: 0.09762859073234723, RMSE: 0.312455742037728
Epoch [16/25], Training Loss: 0.07058386182994582, RMSE: 0.2656762349739732
Epoch [16/25], Validation Loss: 0.07524156538071111, RMSE: 0.274301960220322
Epoch [17/25], Training Loss: 0.07273316423874349, RMSE: 0.26969086791870334
Epoch [17/25], Validation Loss: 0.11067435654462315, RMSE: 0.33267755641855845
Epoch [18/25], Training Loss: 0.07076554689090699, RMSE: 0.26601794467837503
Epoch [18/25], Validation Loss: 0.06755768674542197, RMSE: 0.2599186156192395
Epoch [19/25], Training Loss: 0.06379179494921118, RMSE: 0.25257037623048983
Epoch [19/25], Validation Loss: 0.09689559314865619, RMSE: 0.31128056982191515
Epoch [20/25], Training Loss: 0.06638851045165212, RMSE: 0.25765967952252855
Epoch [20/25], Validation Loss: 0.10572373759932815, RMSE: 0.32515186851581857
Epoch [21/25], Training Loss: 0.06858046575915068, RMSE: 0.26187872338002316
Epoch [21/25], Validation Loss: 0.08053909256588668, RMSE: 0.2837941024156187
Epoch [22/25], Training Loss: 0.06565505595877767, RMSE: 0.25623242565838084
Epoch [22/25], Validation Loss: 0.06414363519288599, RMSE: 0.25326593768781064
Epoch [23/25], Training Loss: 0.06109868009341881, RMSE: 0.247181471986512
Epoch [23/25], Validation Loss: 0.08187964844284579, RMSE: 0.28614620116794454
Epoch [24/25], Training Loss: 0.06072298150276765, RMSE: 0.2464203350025473
Epoch [24/25], Validation Loss: 0.05497483915416524, RMSE: 0.2344671387511803
Epoch [25/25], Training Loss: 0.06492054141592235, RMSE: 0.25479509692284574
Epoch [25/25], Validation Loss: 0.0721349044755334, RMSE: 0.2685794193074618
Finished Training
```

Рисунок 3.12 – Логи тренування та валідації для прогнозу значень за 12 годин

На наступних графіках (рис. 3.13, рис. 3.14) виведено прогнозовані та реальні значення змінних на всьому тестовому наборі.

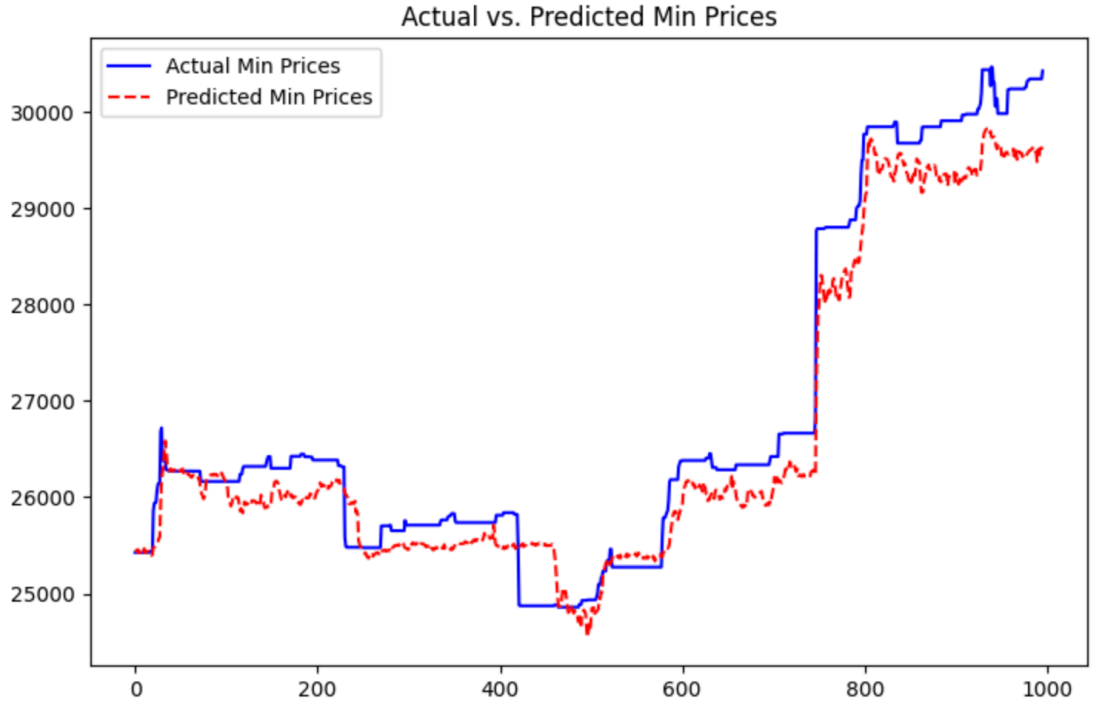

Рисунок 3.13 – Графік мінімальних реальних та прогнозованих значень

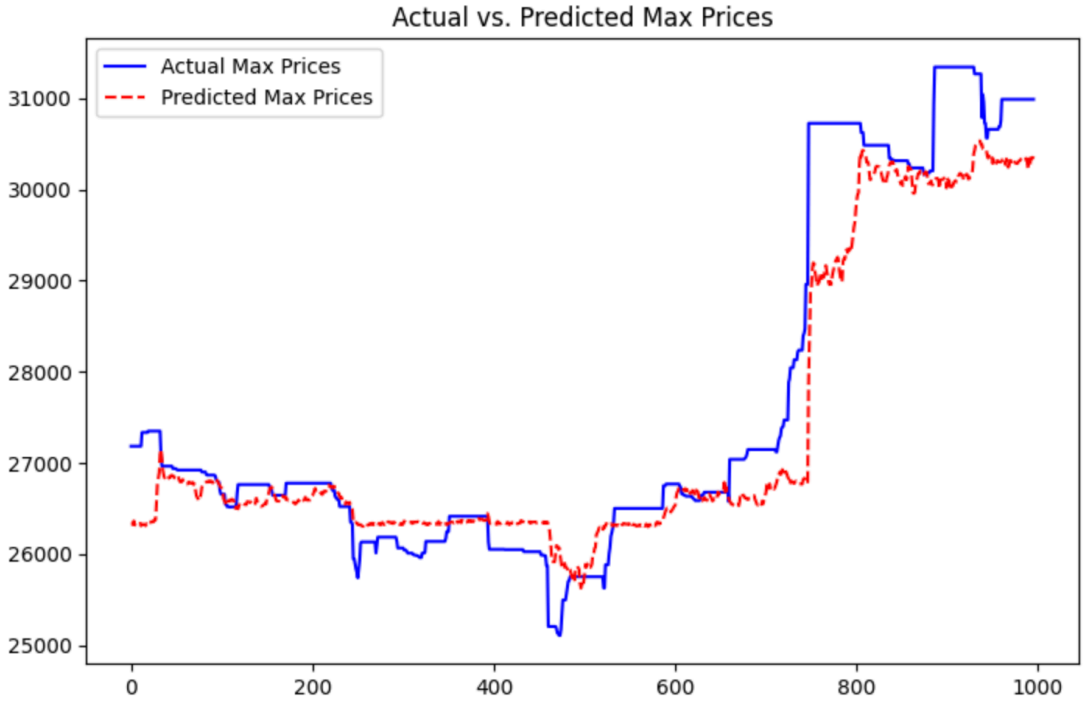

Рисунок 3.14. – Графік максимальних реальних та прогнозованих значень

Результати оцінки якості моделі виведено в таблицю 3.2.

Таблиця 3.2. – Оцінки якості моделі для прогнозу за 12 годин

MSE	RMSE
0.072	0.269

Для прогнозу цих даних модель впоралась гріше, але теж доволі прийнятно. Бачимо, що оцінки якості моделі є трохи гіршими, ніж у попередньому варіанті. Але очевидно, що спрогнозувати максимальні та мінімальні значення за більший проміжок часу (12 годин, а не 1 година) є складнішею задачею. Не забуваймо, що на ціна біткоїна може значно коливатися від впливу багатьох факторів.

3.3 Опис структури програмного продукту

Програмний код поділяється на на такі основні модулі, як модуль обробки та підготовки вхідних даних, модуль навчання та прогнозування, модуль виведення та аналізу результатів.

Було імплементовано 2 основних допоміжних класи. ForecastingModelV2, що наслідує Module від PyTorch. Було перевизначено 2 основних методи класу під задану задачу. CryptoDataset, що наслідує Dataset від PyTorch. Цей клас допомагає зібрати дані для подальшого навчання в цілісну структуру, за заданим шаблоном.

3.4 Мова програмування, середовище розробки і запуску та використані бібліотеки

3.4.1 Мова програмування

Для реалізації програмного продукту було обрано мову програмування Python. Python є однією з найбільш популярних мов для задач машинного навчання завдяки простоті імплементації, читабельності коду та значній підтримці спільноти. Основними перевагами цієї мови є декілька вагомих критеріїв. Перше, це широкий набір бібліотек та фреймворкі, які значно спрощують процес розробки моделей машинного навчання. Друге, підтримка розробницьокю спільною, оскільки мова активно розвивається та вдосконалюєтьс. Також це забезпечує допомогу в рішеннях за вже готовими прикладами та патернами.

3.4.2 Середовище розробки і запуску

Для розробки програмного рішення було обрано Google Colab. Це хмарне середовище на основі Jupyter Notebook, яке надає безкоштовний доступ до графічних процесорів (GPU) та інших обчислювальних ресурсів, які є важливими для навчання моделей. Основними перевагами є безкоштовний доступ до GPU, що значно пришвидшує процес навчання нейронних мереж, хмарне сховище в Google Drive, що дозволяє автоматично зберігати прогрес та мати доступ з будь-якого пристрою через акаунт.

3.4.3 Бібліотеки

Нижче наведено сновні бібліотеки, які було використано в програмному продукті.

Бібліотека PyTorch використовує динамічні обчислювальні графи, що дозволяє змінювати архітектуру моделі під час навчання. Це робить PyTorch гнучким та зручним для досліджень та експериментів. Крім того, він має хорошу документацію та активну спільноту. Значною перевагою PyTorch є проста інтеграція з CUDA, що дозволяє ефективно використовувати GPU для прискорення обчислень.

Бібліотека Transformers (від Hugging Face) пропонує готові до використання моделі для різноманітних задач обробки природної мови (NLP), такі як BERT, GPT-3, RoBERTa тощо. Це дозволяє швидко адаптувати та використовувати готові доробки у сфері NLP. Бібліотека має зручний інтерфейс та легко інтегрується з PyTorch, що дозволяє користуватися різними класами моделей з мінімальними зусиллями.

Бібліотека Matplotlib забезпечує візуалізацію даних. Вона дозволяє відмальовувати різноманітні типи графіків, гістограм, діаграм тощо. Значною

перевагою є налаштування візуальних деталей, включаючно з кольорами, шрифтами, підписами та іншим, що забезпечує читабельність візуалізованих даних. Інтеграція з бібліотеками NumPy та Pandas дозволяє легко створювати графіки на основі попередньо оброблених даних.

Бібліотека Scikit-learn є зручною бібліотекою для машинного навчання, яка пропонує значну кількість алгоритмів для класифікації, регресії, кластеризації та зменшення розмірності даних, розбиття даних тощо. Scikit-learn має детальну документацію та велику спільноту користувачів, що забезпечує доступ до ресурсів для навчання та вирішення проблем.

Бібліотека Pandas є однією з бібліотек для обробки та аналізу даних. Вона забезпечує ефективну роботу з датафреймами та дозволяє виконувати складні операції над ними, включаючи фільтрацію, групування та агрегацію даних. Обробка таких даних не вимагає складного коду, адже Pandas пропонує прості і зрозумілі методи. Pandas добре інтегрується з іншими бібліотеками, що забезпечує комплексний підхід до аналізу та обробки даних.

Бібліотека NumPy є фундаментальною бібліотекою для обчислень у Python, яка забезпечує ефективну роботу з багатовимірними масивами та матрицями. Вона містить велику кількість функцій для виконання математичних та статистичних обчислень. Також дозволяє виконувати векторизовані операції, що може пришвидшити обробку даних у порівнянні з традиційними методами із циклами.

3.5 Ідеї покращення розробленого програмного продукту

3.5.1 Написання юзер-френдлі інтерфейсу

У даній роботі представлено основний функціонал, бекенд частину для прогнозування курсу криптовалюти біткоїн на основі аналізу новин. Проте у наш

час, зважаючи на конкуренцію на ринку, таким підходом користувачів важко здивувати, та й загалом заработи на цьому гроші. Зважаючи на те, що в такому функціоналі мають потребу в основному звичайні користувачі, а не вузьконаправлені спеціалісти з розробки, наприклад, існує гостра потреба в доступному та зрозумілому відображенні результатів. Інтерфейс може реалізовувати відображення прогнозу, тенденції спаду або зросту курсу, новини, які впливають на курс.

3.5.2 Прогнозування у реальному часі

Корисним функціоналом може стати модуль, який буде збирати актуальні новини в режимі реального часу та виводити динамічний прогноз на основі отриманих даних. Задача збору даних є значним викликом та окремим великим полем дослідження, адже потрібно збирати з багатьох джерел та фільтрувати велику кількість інформації, зважаючи на актуальність змісту для даної задачі. Тут можуть бути застосовані різні допоміжні мовні моделі, наприклад, які ймовірно гарно впораються із цією задачею. Проте така модифікація потребує великої кількості ресурсів та є дуже комплексною.

3.5.3 Розширення прогнозу на інші види криптовалют

Створений програмний продукт застосовується для прогнозування курсу біткоїна, проте існує ввелика кількість інших криптографічних валют, що займають значні частину фінансового ринку. Користувачам важливо бачити прогнозування на значну кількість популярних криптовалют, адже це може послужити значним важелем у прийнятті рішень щодо їхньої купівлі та продажу, продемонструє, куди

краще вкладати гроші, а де збільшують ризики втрат тощо. Тому розширення прогнозу на інші валюти є значною корисною функцією та вагомою перевагою серед конкурентного ринку.

Висновки до розділу 3

У третьому розділі було наведено практичну реалізацію рішення даної задачі прогнозування курсу біткоїна, проведено аналіз отриманих результатів. Було наведено побудову моделі, опис модулів програмного продукту та використані інструменти.

Результати роботи програмного продукту є доволі показовими та свідчать про успішну роботу моделі. Основною задачею було здійснити не точний прогноз значень курсу біткоїна, а відслідкувати динаміку зміни, аналізуючи актуальні новини. Видно, що вплив новин на курс біткоїна є значним фактором, тому на їхній основі вдалось навчити розроблену модель.

Прогнозування динаміки зростання чи спадання курсу біткоїна важливіше, ніж прогнозування точного значення з кількох ключових причин.

По-перше, ринок криптовалют надзвичайно волатильний і прогнозувати точні значення курсів дуже складно через безліч факторів, які на них впливають.

По-друге, прийняття інвестиційних рішень та побудова торгівельних стратегій потребують не точних значень, а розуміння загального напрямку руху ринку. Новини висвітлюють актуальну ситуацію у світі, що значно впливає на фінансовий ринок і його динаміку.

Розроблений програмний продукт не є довершеною версією, наприклад, через брак матеріальних ресурсів, тому було наведено ідеї з покращення програми, щоб бути конкурентноспроможними та надавати користувачам максимально корисний функціонал.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

Цей розділ включає проведення оцінювання основних характеристик для майбутнього програмного продукту, що спеціалізується на дослідженні демографічного стану.

Дана реалізація сприятиме проведенню усіх необхідних досліджень, що нададуть змогу якісно дослідити питання не лише в Україні, проте ще й у всьому світі.

Також в даному дослідженні показано різні варіанти реалізації для забезпечення найбільш коректної та оптимальної стратегії вибору, що має вплив на економічні фактори та сумісність з майбутнім програмним продуктом. Для цього застосовувався апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) передбачає собою технологію, що дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення резервів зниження витрат за рахунок ефективніших варіантів виробництва, кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

4.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи прогнозу стійкості фінансових показників. Оскільки рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу даних по компанії.

Технічні вимоги до програмного продукту є наступні:

1) функціонування на персональних комп'ютерах із стандартним набором компонентів;
2) зручність та зрозумілість для користувача;
3) швидкість обробки даних та доступ до інформації в реальному часі;
4) можливість зручного масштабування та обслуговування;
5) мінімальні витрати на впровадження програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка можливого програмного продукту, яка дозволяє аналізувати різні характеристики, що безпосередньо впливають на стійкість підприємства. Беручи за основу цю функцію, можна виділити наступні:

F_1 – вибір мови програмування.

F_2 – вибір реалізації базових функцій.

F_3 – вибір програмного середовища розробки продукту.

Кожна з цих функцій має декілька варіантів реалізації:

1) функція F_1:

а) Python;

б) R.

2) функція F_2:

а) Застосування вбудованих функцій та допоміжних бібліотек;

б) Створення своїх функцій.

3) Функція F_3:

а) Jupyter Notebook;

б) VS Code.

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1).

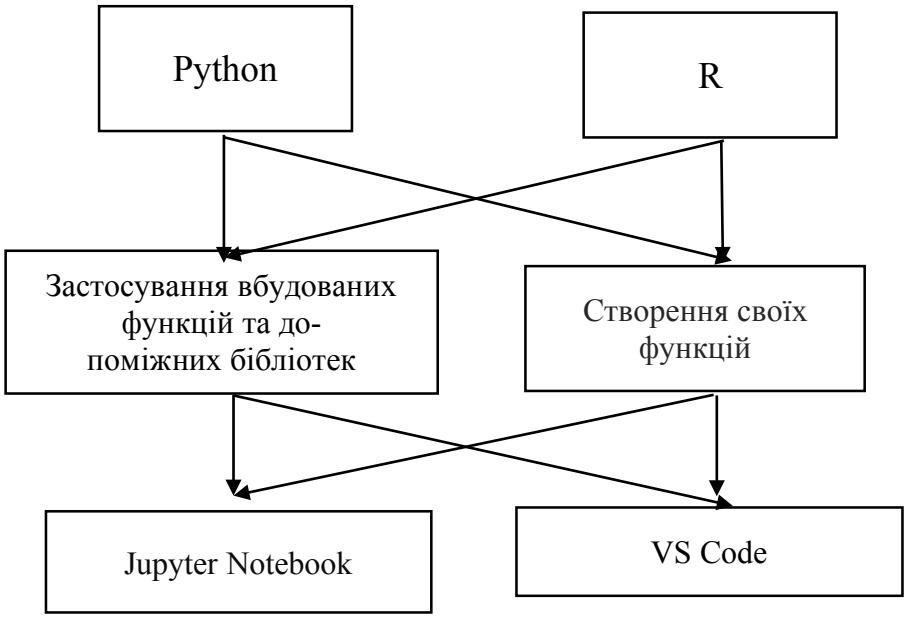

Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантиів основних функцій.

Позитивно-негативна матриця показана в таблиці 4.1.

Таблиця 4.1 – Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	*А*	Велика кількість бібліотек для аналізу, обробки даних та навчання, простота використання	Може бути менш продуктивним за інші мови програмування
	Б	Орієнтована на статистику та аналіз даних, з великою кількістю вбудованих функцій	Менш універсальна в порівнянні з Python, складніша у використанні
F_2	*А*	Висока швидкість розробки, надійність та підтримка використання	Обмеження стандартними можливостями
	Б	Гнучкість та повний контроль над поведінкою	Потребує більше ресурсів для розробки, необхідність тестування
F_3	*А*	Доступність та легкість при написанні, інтерактивність	Менш зручний для розробки великих проєктів, знижена продуктивність
	Б	Багатофункціональне, зручне управління проєктом, потужні інструменти розробки	Потребує додаткових налаштувань та конфігурацій, складність інтерфейсу

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F_1:

Перевагу даємо загальнодоступності та простоті у реалізації. Для спрощення роботи по написанню коду варіант Б має бути відкинутий.

Функція F_2:

Реалізація першого варіанту є сприйнятливою для поставленої задачі. Це варіант А.

Функція F_3:

Програма допускає обрання обох варіантів. Можливо використати варіанти А чи Б.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

F_1а – F_2а – F_3а

F_1а – F_2а – F_3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів програмного продукту

На основі даних, розглянутих вище, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- *X1* – швидкодія мови програмування;
- *X2* – об’єм оперативної пам’яті;
- *X3* – час попередньої обробки даних;
- *X4* – потенційний об’єм програмного коду.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту, як показано у таблиці 4.2.

Таблиця 4.2 – Основні параметри програмного продукту

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X1	оп/мс	9000	15000	20000
Об’єм ОП	X2	Мб	8	16	32
Час попередньої обробки даних	X3	с	100	80	70
Потенційний об’єм програмного коду	X4	кількість рядків коду	1500	1200	1000

За даними таблиці 4.3 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

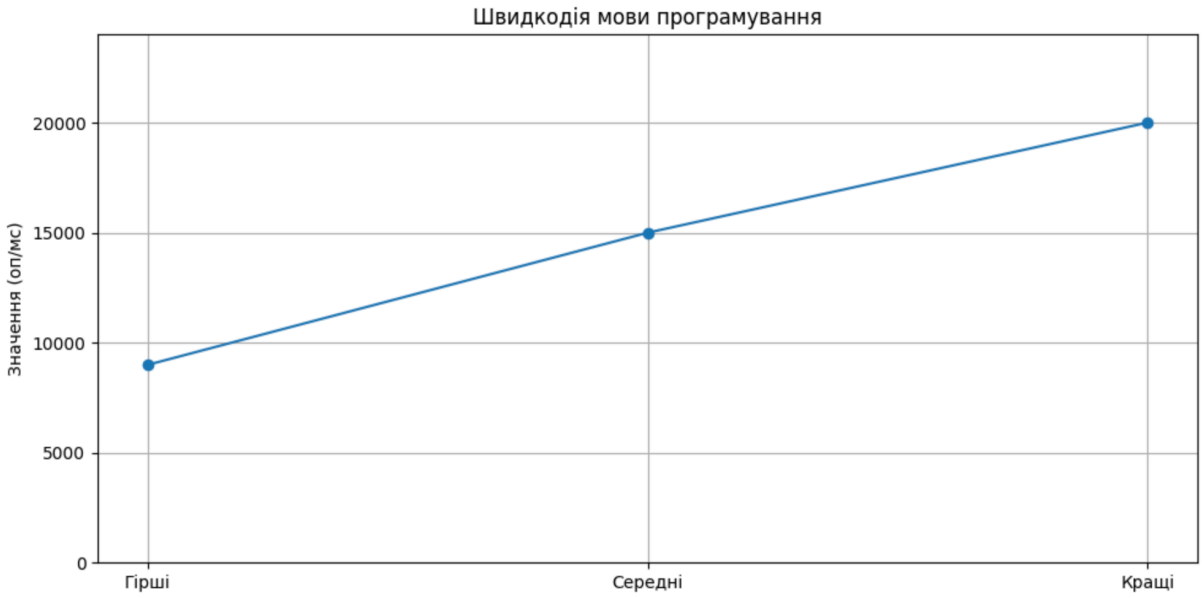

Рисунок 4.2 – X1, швидкодія мови програмування

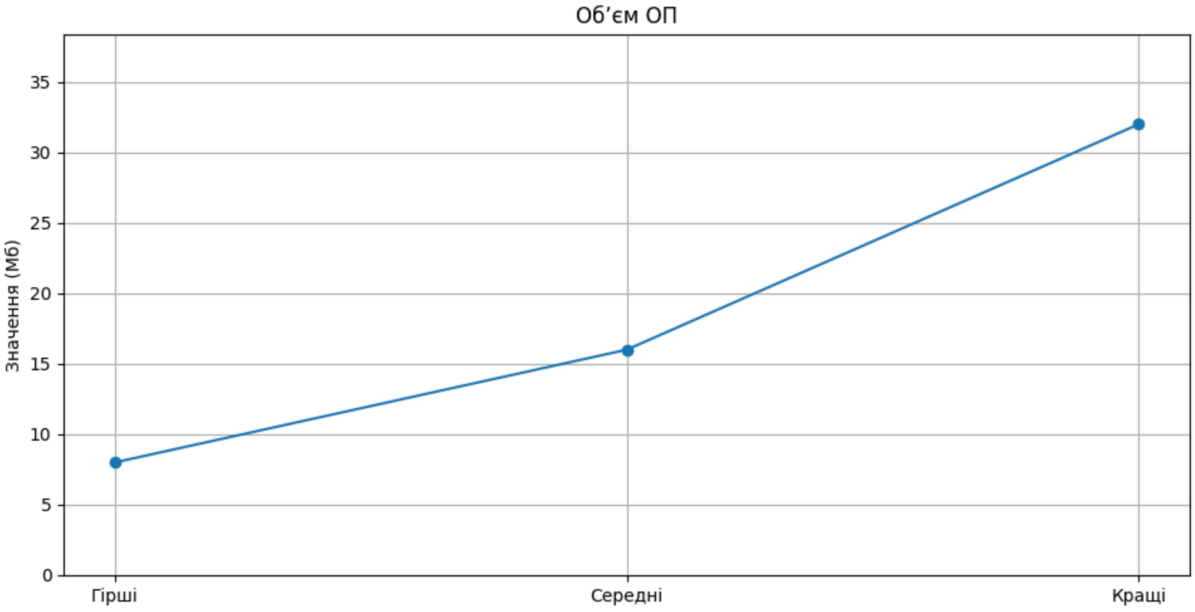

Рисунок 4.3 – X2, об’єм пам’яті

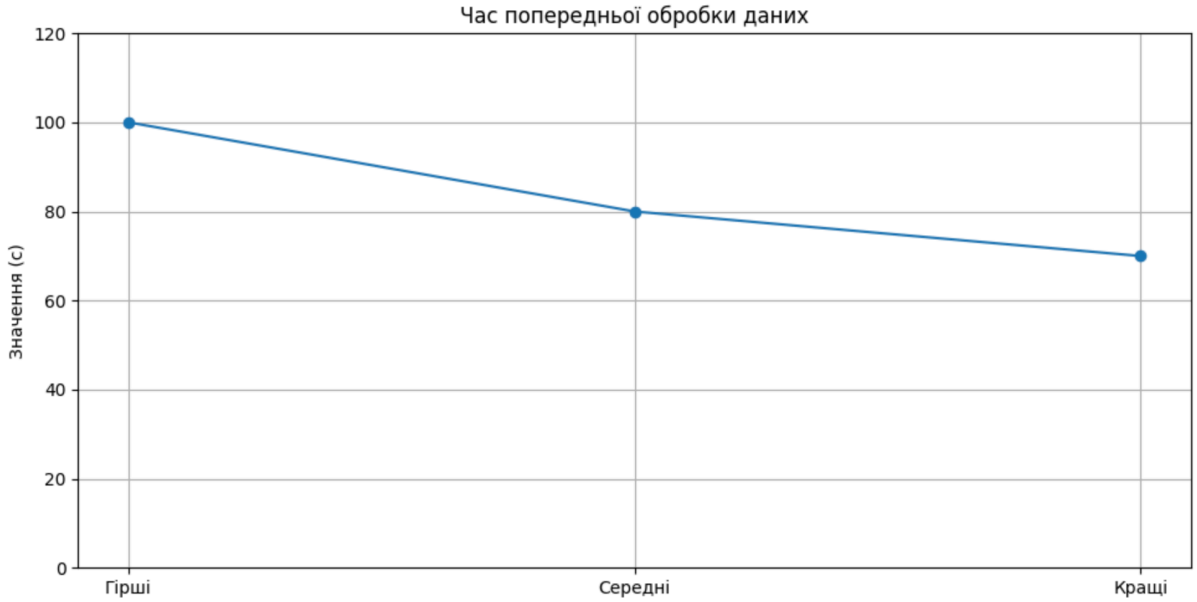

Рисунок 4.4 – X3, час попередньої обробки даних

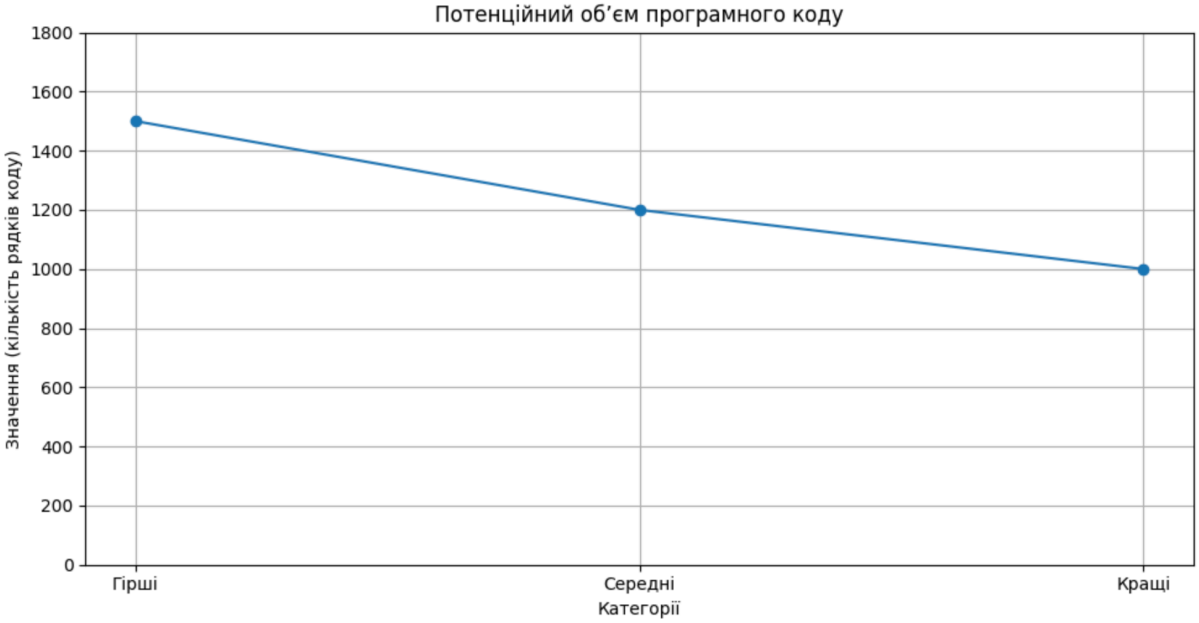

Рисунок 4.5 – X4, потенційний об’єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка

програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого використання;
- визначення оцінки попарного пріоритету параметрів;
- обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхи-лення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	1	2	2	2	2	1	2	12	-5,5	30,25
X2	Об’єм ОП	Mb	2	1	1	1	1	2	1	9	-8,5	72,25
X3	Час попередньої обробки даних	мс	4	4	3	3	3	3	4	24	6,5	42,25
X4	Потенційний об’єм програмного коду	Кількість рядків коду	3	3	4	4	4	4	3	25	7,5	56,25
	Разом		10	10	10	10	10	10	10	70	0	201

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70, \quad (4.1)$$

де N – число експертів,

n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17{,}5 \quad (4.2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T. \quad (4.3)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 201. \quad (4.4)$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 201}{7^2(4^3 - 4)} = 0{,}82 > W_k = 0{,}67. \quad (4.5)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 – Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	<	>	>	>	>	<	>	>	1,5
X1 і X3	<	<	<	<	<	<	<	<	0,5
X1 і X4	<	<	<	<	<	<	<	<	0,5
X2 і X3	<	<	<	<	<	<	<	<	0,5
X2 і X4	<	<	<	<	<	<	<	<	0,5
X3 і X4	>	>	<	<	<	<	>	<	0,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}. \quad (4.6)$$

З отриманих числових оцінок переваги складемо матрицю A=$\| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{ві}$ за наступними формулами:

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i} \quad (4.7)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \quad (4.8)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \tag{4.9}$$

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j \tag{4.10}$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$
X1	1	1,5	0,5	0,5	3,5	0,22	12,25	0,2
X2	0,5	1	0,5	0,5	2,5	0,16	9,25	0,16
X3	1,5	1,5	1	0,5	4,5	0,28	16,25	0,28
X4	1,5	1,5	1,5	1	5,5	0,34	21,25	0,36
Всього:					16	1	59	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2* (Об’єм пам’яті), *X3* (час попередньої обробки даних) та *X4* (потенційний об’єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X1* (швидкість роботи мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j}\ , \qquad (4.11)$$

де n – кількість параметрів;

$K_{вi}$– коефіцієнт вагомості i–го параметра;

B_i – оцінка i–го параметра в балах.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	А	X1	11000	10	0,2	2
F3	А	X2	16	14	0,16	2,24
	Б	X3	80	12	0,28	3,36
F2	А	X4	1000	6	0,36	2,16

За даними з таблиці 4.6 за формулою:

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}], \qquad (4.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 2 + 2{,}24 + 2{,}16 = 6{,}4;$$

$$K_{K2} = 2 + 3{,}36 + 2{,}16 = 7{,}52.$$

Як видно з розрахунків, кращим є 2 варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Теоретичне проектування програмного продукту;

2. Розробка програми;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як:

$$\mathrm{T_O} = \mathrm{T_P} \cdot \mathrm{K_П} \cdot \mathrm{K_{СК}} \cdot \mathrm{K_М} \cdot \mathrm{K_{СТ}} \cdot \mathrm{K_{СТ.М}}, \qquad (4.13)$$

де $\mathrm{T_P}$ – трудомісткість розробки ПП;

$\mathrm{K_П}$ – поправочний коефіцієнт;

$\mathrm{K_{СК}}$ – коефіцієнт на складність вхідної інформації;

$\mathrm{K_М}$ – коефіцієнт рівня мови програмування;

$К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $Т_Р$ =50 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.9. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$Т_1 = 50 \cdot 1.7 \cdot 0.9 = 76,5 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р$ =30 людино-днів, $К_П$ =0.8, $К_{СК}$ = 1, $К_{СТ}$ =0.8:

$$Т_2 = 30 \cdot 0.8 \cdot 0.8 = 19,2 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$Т_I = (76,5 + 19,2 + 5.2 + 19,2) \cdot 8 = 960,8 \text{ людино-годин.}$$

$$Т_{II} = (76,5 + 19,2 + 7.8 + 19,2) \cdot 8 = 981,6 \text{ людино-годин.}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 25000 грн., один аналітик в області даних з окладом 20000. Визначимо середню зарплату за годину за формулою:

$$С_ч = \frac{М}{T_m \cdot t} грн., \quad (4.14)$$

де М – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

$$С_ч = \frac{25000 + 25000 + 20000}{3 \cdot 21 \cdot 8} = 138{,}89 \ \ грн. \quad (4.15)$$

Тоді, розрахуємо заробітну плату за формулою:

$$С_{ЗП} = С_ч \cdot Т_i \cdot К_Д, \quad (4.16)$$

де $С_ч$– величина погодинної оплати праці програміста;

$Т_i$ – трудомісткість відповідного завдання;

$К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 138{,}89 \cdot 960{,}8 \cdot 1.2 = 160134{,}61$ грн.

II. $С_{ЗП} = 138{,}89 \cdot 981{,}6 \cdot 1.2 = 163601{,}31$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 160134{,}61 \cdot 0.22 = 35229{,}61$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 163601{,}31 \cdot 0.22 = 35992{,}29$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 25000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 25000 \cdot 0,2 = 60000 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 60000 \cdot (1 + 0.2) = 72000 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 72000 \cdot 0,22 = 15840 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 15% та вартості ЕОМ – 10000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.3 \cdot 0.15 \cdot 10000 = 1950 \text{ грн.,}$$

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.4 \cdot 10000 \cdot 0.08 = 1120 \text{ грн.,}$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$T_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (366 – 104 – 0 – 25) \cdot 8 \cdot 0.45 =$$

$$= 853,2 \text{ години},$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = T_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 853,2 \cdot 0,2 \cdot 0,3 \cdot 5,21 = 266,7 \text{ грн.},$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 10000 \cdot 0,67 = 6700 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н, \quad (4.17)$$

$$С_{ЕКС} = 72000 + 15840 + 1950 + 1120 + 266,7 + 6700 = 97876,70 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 97876,70 / 853,2 = 114,72 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М-Г} \cdot Т, \quad (4.18)$$

I. $С_М = 114,72 \cdot 960,8 = 110222,98$ грн.

II. $С_М = 114,72 \cdot 981,6 = 112609,15$ грн.

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0,67, \quad (4.19)$$

I. $С_Н = 160134,61 \cdot 0,67 = 107290,19$ грн.

II. $С_Н = 163601,31 \cdot 0,67 = 109612,88$ грн.

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н, \quad (4.20)$$

I. $С_{ПП} = 160134,61 + 35229,61 + 110222,98 + 107290,19 = 412877,39$ грн.

II. $С_{ПП} = 163601,31 + 35992,29 + 112609,15 + 109612,88 = 421815,63$ грн.

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР j} = К_{Кj} / С_{Фj}, \quad (4.21)$$

$$К_{ТЕР1} = 6,4 / 412877,39 = 1,55 \cdot 10^{-5},$$

$$К_{ТЕР2} = 7,52 / 421815,63 = 1,78 \cdot 10^{-5}.$$

Як бачимо, найбільш ефективним є другий варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 1,78 \cdot 10^{-5}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є другий варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 1,78 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

- Вибір мови програмування – Python;
- Реалізація важливої постановки з допомогою вбудованих функцій та допоміжних бібліотек;
- Використання середовища розробки VS Code.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, швидку реалізацію програми та доступний функціонал для роботи.

Висновки до розділу 4

В даній частині було проведено повний функціонально-вартісний аналіз програмного продукту. Також було знайдено оцінку основних функцій програмного продукту.

В результаті виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, було визначено та проведено оцінку основних функцій програмного продукту, а також знайдено параметри, які його характеризують.

На основі аналізу вибрано варіант реалізації програмного продукту.

ВИСНОВКИ

У результаті виконання даної роботи було розроблено модель для прогнозування курсу криптовалюти біткоїн на основі інформаційних збурень, із застосуванням лінгвістичної моделі BERT та рекурентної нейронної мережі LSTM. Було використано дану модель на реальних даних для прогнозування мінімальних та максимальних значень курсу через 1 годину та 12 годин.

У першому розділі було вивчено предметну область. Було розглянуто актуальність цієї проблеми, досліджено історію розвитку криптовалют. Зроблено висновок, що криптографічний ринок є вагомою частиною фінансового економічного ринку, а також активно розвивається на поширюється по всьому світу. Розглядалися існуючі комерційні підходи до розв’язання задачі прогнозу криптовалют, але ці методи лишаються для звичайного користувача «чорною коробкою».

У другому розділі було розглянуто теоретичні відомості та методики для побудови моделі прогнозування курсу криптовалют. Зроблено висновок, що в даній задачі треба будувати нейронну мережу, із використанням лінгвістичної моделі BERT та рекурентної нейронної мережі LSTM. Детальніше розглянуто основну ідею цих розробок та функціонування. Згадано підхід приведення даних до одного виду та метрики оцінок якості моделі.

У третьому розділі було описано деталі реалізації програмного продукту. Описано побудову моделі. Розписано відомості про вхідний датасет та роботу з ним для подальшого навчання та прогнозування. Згадано використані технології та бібліотеки для реалізації програми та наголошено на їхніх перевагах. Висвітлено результати роботи програми з отриманими значення метрик якості моделі. Наведено ідеї покращення та розширення програмного продукту.

У четвертому розділі було виконано функціонально-вартісний аналіз розробленого програмного продукту.

Загалом можна зробити висновок, що у ході виконання цього дослідження було досягнуто поставленої мети цієї роботи, а саме розробки моделі прогнозування курсу криптовалюти на основі інформаційних збурень, проаналізувавши їхній вплив на курс біткоїну. Основна частина та база для задачі прогнозування курсу біткоїна була виконана. Проте є ще шляхи допрацювання цього продукту, щоб він став конкурентноспроможним у сфері прогнозування курсу криптовалют і зміг приносити значний дохід. Наприклад, покращення точності прогнозу за рахунок врахування більшої кількості факторів. Також розробка зручного для користувачів інтерфейсу, що висвітлює провідні новини, а відповідно до них динамічний прогноз курсу біткоїна. І наостанок розширення переліку можливих криптографічних валют, для яких буде здійснюватися прогноз.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Федорова Ю. Криптовалюти та їх місце у фінансовій системі. *Економіка і суспільство*. 2018. № 15.
URL: https://economyandsociety.in.ua/journals/15_ukr/116.pdf. (дата звернення: 10.05.2024).
2. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. URL: https://git.dhimmel.com/bitcoin-whitepaper (Last accessed: 10.05.2024).
3. Luther W. J. The rise of bitcoin. *AIER*. URL: https://www.aier.org/article/the-rise-of-bitcoin/?gad_source=1&gclid=Cj0KCQjwxeyxBhC7ARIsAC7dS3-12dPZS2fE4LoVAd9m_Crvrc4AZ05F6bTozn YpsKmOFqDYLvlchVoaAsBOEALw_wcB (Last accessed: 10.05.2024).
4. Бабінська С. Технологія блокчейн в аудиті: сучасний стан та перспективи застосування. *Економіка та суспільство*. 2022. № 36.
5. Lapko O. O., Solosich O. S. Blockchain technology: the concept, scope and impact on business. *Business inform*. 2019. Vol. 6, № 497. P. 77–82.
6. Литвиненко А. О. Вплив криптовалюти на світову економіку / А. О. Литвиненко, В. І. Ситніков. *Управління ресурсним забезпеченням господарської діяльності підприємства реального сектору економіки* : матер. VI Всеукр. наук.-практ. інтернет-конф. з міжнар. участю : м. Полтава, 17 лист. 2021 р. : тези допов. – Полтава: НДАУ, 2021. – С. 51-54, URL: http://repository.hneu.edu.ua/handle/123456789/26916 (дата звернення: 10.05.2024).
7. Editorial C. Best AI tools for crypto prices predictions. Medium. URL: https://medium.com/@cryptonicaed/best-ai-tools-for-crypto-prices-predictions-1578919d4e25 (Last accessed: 10.05.2024).
8. High-Level Expert Group on Artificial Intelligence set up by the European Commission. A definition of AI: Main capabilities and scientific disciplines. 2019.

URL: https://ec.europa.eu/futurium/en/system/files/ged/ai_hleg_definition_of_ai_18_december_1.pdf (Last accessed: 10.05.2024).

9. Mahesh B. Machine Learning Algorithms - A Review. *International Journal of Science and Research (IJSR)* URL: https://www.researchgate.net/publication/344717762_Machine_Learning_Algorithms_-A_Review (Last accessed: 10.05.2024).

10. Dreyfus S. E. Artificial neural networks, back propagation, and the Kelley-Bryson gradient procedure. *Journal of Guidance, Control, and Dynamics*. 1990.

11. LeCun Y., Bengio Y., Hinton G. Deep learning. *Nature*. 2015. Vol. 521, № 7553. P. 436–444.

12. Bidyuk P., Huts Y., Gavrilenko V., Rudoman N. Прогнозування цін акцій з використанням рекурентної нейронної мережі lstm. *Системи управління, навігації та зв'язку. Збірник наукових праць*. 2021. Т. 3, № 65. С. 64–68.

13. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin I. Attention is All you Need. *Advances in Neural Information Processing Systems.* № 30. 2017.

14. Bahdanau Ch., Kyunghyun B., Yoshua. Neural Machine Translation by Jointly Learning to Align and Translate. 2014.

15. Hochreiter S., Schmidhuber J. Long Short-Term Memory. *Neural computation*. 1997. Vol. 9, № 8. P. 1735–1780.

16. Devlin J., Chang M.-W. Open sourcing BERT: state-of-the-art pre-training for natural language processing. URL: https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html (Last accessed: 10.05.2024).

17. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. URL: https://web.archive.org/web/20201230043015/https://arxiv.org/pdf/1810.04805.pdf (Last accessed: 10.05.2024).

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

```
import pandas as pd
import numpy as np
import torch
from torch import nn, optim
from torch.nn import functional as F
from torch.utils.data import Dataset, DataLoader
from transformers import BertTokenizer, BertModel
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from google.colab import drive

# get dataset from file system
drive.mount('/content/drive')
price_after_news_first_df = pd.read_csv('drive/MyDrive/диплом/price_after_news.csv')
price_after_news_first_df.head(5)
price_after_news_df = price_after_news_first_df.drop(columns=['link', 'symbol', 'index_price', 'interest',
'24hour_future_price_min', '24hour_future_price_max'], axis=1)
price_after_news_df.count()

#func to modify tags
price_after_news_df['tags'] = price_after_news_df['tags'].fillna(2)
def normalise_row(row):
    if row['tags'] == 2:
      return 2
    elif "BTC" in row['tags']:
      return 1
    else:
      return 0

price_after_news_df['tags'] = price_after_news_df.apply(lambda row : normalise_row(row), axis=1)
price_after_news_df.count()
price_after_news_df['tags'].hist()

# Visualizing a sample of predictions
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(15, 5))
plt.subplot(1, 2, 1)
plt.plot(price_after_news_df['mark_price'], label="mark price", color="blue")
plt.legend()

plt.tight_layout()
plt.show()

cols_to_convert = ['mark_price','funding', 'tags']
n = len(cols_to_convert)

# convert to float64
price_after_news_df[cols_to_convert] = price_after_news_df[cols_to_convert].apply(pd.to_numeric,
errors='coerce')

price_after_news_df_copy = price_after_news_df
first_train_items_count = int(len(price_after_news_df['mark_price']) * 0.8)
print(first_train_items_count)

# get all texts
text_columns = ['caption', 'short_content']
pred_columns = ['1hour_future_price_min','1hour_future_price_max']

# concat caption with content func
def concat_col(values):
  return f"caption: {values['caption']}; content: {values['short_content']}"

SEQUENCE_LENGTH = 10
scaler = StandardScaler()
targets, texts = [], []
for row in price_after_news_df.iterrows():
    texts.append(concat_col(row[1]))

texts = texts[SEQUENCE_LENGTH-1:]

# standartization
```

```
scaler.fit(price_after_news_df['1hour_future_price_min'].values[:].reshape(-1, 1))
print(scaler.fit_transform(price_after_news_df['1hour_future_price_min'].values[:].reshape(-1, 1)).reshape(1, -1)[0])
price_after_news_df['1hour_future_price_min'] =
scaler.fit_transform(price_after_news_df['1hour_future_price_min'].values[:].reshape(-1, 1)).reshape(1, -1)[0]
hour_future_price_min_mean, hour_future_price_min_std = scaler.mean_, scaler.scale_

scaler.fit(price_after_news_df['1hour_future_price_max'].values[:].reshape(-1, 1))
price_after_news_df['1hour_future_price_max'] =
scaler.fit_transform(price_after_news_df['1hour_future_price_max'].values[:].reshape(-1, 1)).reshape(1, -1)[0]
hour_future_price_max_mean, hour_future_price_max_std = scaler.mean_, scaler.scale_

targets = price_after_news_df[['1hour_future_price_min',
'1hour_future_price_max']].values[SEQUENCE_LENGTH-1:]

scaler.fit(price_after_news_df['mark_price'].values[SEQUENCE_LENGTH-1:].reshape(-1, 1))
price_after_news_df['mark_price'] = scaler.fit_transform(price_after_news_df['mark_price'].values[:].reshape(-1, 1)).reshape(1, -1)[0]
scaler.fit(price_after_news_df['estimated_settle_price'].values[SEQUENCE_LENGTH-1:].reshape(-1, 1))
price_after_news_df['estimated_settle_price'] =
scaler.fit_transform(price_after_news_df['estimated_settle_price'].values[:].reshape(-1, 1)).reshape(1, -1)[0]

# get all TS data
time_series_data = []
for i in range(len(price_after_news_df) - SEQUENCE_LENGTH + 1):
    time_series_data.append(price_after_news_df[cols_to_convert].iloc[i:i+SEQUENCE_LENGTH].values)

time_series_data = np.array(time_series_data)

# train/test split 80%20
texts_train, texts_test = texts[:first_train_items_count], texts[first_train_items_count:]
print(len(texts_test))
ts_data_train, ts_data_test = time_series_data[:first_train_items_count],
time_series_data[first_train_items_count:]
print(len(ts_data_test))
targets_train, targets_test = targets[:first_train_items_count], targets[first_train_items_count:]
print(len(targets_test))
```

```
from transformers import BertModel, BertTokenizer

tokenizer = BertTokenizer.from_pretrained('bert-base-cased')
bert_model = BertModel.from_pretrained('bert-base-cased')

class CryptoDataset(Dataset):
    def __init__(self, texts, time_series_data, targets, tokenizer, max_len):
        self.texts = texts
        self.time_series_data = time_series_data
        self.targets = targets
        self.tokenizer = tokenizer
        self.max_len = max_len

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        text = str(self.texts[idx])
        inputs = self.tokenizer.encode_plus(
            text,
            None,
            add_special_tokens=True,
            max_length=self.max_len,
            padding='max_length',
            return_token_type_ids=True,
            truncation=True
        )
        input_ids = inputs['input_ids']
        attention_mask = inputs['attention_mask']
        return {
            'input_ids': torch.tensor(input_ids, dtype=torch.long),
            'attention_mask': torch.tensor(attention_mask, dtype=torch.long),
            'time_series': torch.tensor(self.time_series_data[idx], dtype=torch.float),
            'targets': torch.tensor(self.targets[idx], dtype=torch.float)
        }

train_dataset = CryptoDataset(texts=texts_train, time_series_data=ts_data_train, targets=targets_train,
tokenizer=tokenizer, max_len=256)
```

```
test_dataset = CryptoDataset(texts=texts_test, time_series_data=ts_data_test, targets=targets_test,
tokenizer=tokenizer, max_len=256)
train_loader = DataLoader(train_dataset, batch_size=10, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=10, shuffle=False)

sample_batch = next(iter(train_loader))
print(sample_batch['input_ids'].shape)
print(sample_batch['attention_mask'].shape)
print(sample_batch['time_series'].shape)
print(sample_batch['targets'].shape)

class ForecastingModelV2(nn.Module):
   def __init__(self, bert_embedding_dim=768, time_series_feature_dim=n, lstm_units=128,
dense_hidden_units=64):
      super(ForecastingModelV2, self).__init__()

      self.bert = bert_model
      for param in self.bert.parameters():
         param.requires_grad = False

      self.fc1 = nn.Linear(bert_embedding_dim, dense_hidden_units)
      self.bn1 = nn.BatchNorm1d(num_features=dense_hidden_units)
      self.relu = nn.ReLU()
      self.dropout = nn.Dropout(p=0.5)

      self.lstm1 = nn.LSTM(input_size=time_series_feature_dim, hidden_size=lstm_units, batch_first=True,
bidirectional=True)
      self.lstm2 = nn.LSTM(input_size=2*lstm_units, hidden_size=lstm_units, batch_first=True,
bidirectional=True)

      # Adjusted input size for fc2
      self.fc2_input_dim = 2*lstm_units + 2*lstm_units + dense_hidden_units
      self.fc2 = nn.Linear(self.fc2_input_dim, dense_hidden_units)
      self.bn2 = nn.BatchNorm1d(num_features=dense_hidden_units)

      self.fc3 = nn.Linear(dense_hidden_units, dense_hidden_units)

      self.fc_min = nn.Linear(dense_hidden_units, 1)
```

```
        self.fc_max = nn.Linear(dense_hidden_units, 1)

    def forward(self, input_ids, attention_mask, time_series_data):
        bert_embeddings = self.bert(input_ids=input_ids, attention_mask=attention_mask).pooler_output
        x_bert = self.fc1(bert_embeddings)
        x_bert = self.bn1(x_bert)
        x_bert = self.relu(x_bert)

        x_time_series, _ = self.lstm1(time_series_data)
        x_time_series, _ = self.lstm2(x_time_series)

        global_max_pooling, _ = torch.max(x_time_series, dim=1)
        global_avg_pooling = torch.mean(x_time_series, dim=1)
        x = torch.cat((global_max_pooling, global_avg_pooling, x_bert), dim=1)

        x = self.fc2(x)
        x = self.bn2(x)
        x = self.relu(x)
        x = self.dropout(x)
        x = self.fc3(x)
        x = self.relu(x)

        min_price = self.fc_min(x)
        max_price = self.fc_max(x)

        return min_price, max_price

# connect gpu
if torch.cuda.is_available():
    mps_device = torch.device("cuda")
    print("Using GPU.")
else:
    print("No GPU available, using the CPU instead.")
    mps_device = torch.device("cpu")

import math
model_v2 = ForecastingModelV2().to(mps_device)
optimizer = optim.Adam(model_v2.parameters(), lr=0.001, weight_decay=1e-5)
```

```
criterion = nn.MSELoss()


EPOCHS = 30
for epoch in range(EPOCHS):
    # Training loop
    model_v2.train()
    running_loss = 0.0
    rmse_sum = 0.0

    for i, data in enumerate(train_loader, 0):
        input_ids = data['input_ids'].to(mps_device)
        attention_mask = data['attention_mask'].to(mps_device)
        time_series = data['time_series'].to(mps_device)
        targets = data['targets'].to(mps_device)

        optimizer.zero_grad()
        outputs = model_v2(input_ids, attention_mask, time_series)
        min_price, max_price = outputs
        outputs_stacked = torch.cat((min_price, max_price), dim=1)
        loss = criterion(outputs_stacked, targets)

        loss.backward()
        optimizer.step()
        running_loss += loss.item()

    training_loss = running_loss / (i+1)

    print(f'Epoch [{epoch + 1}/{EPOCHS}], Training Loss: {running_loss / (i+1)}, RMSE:
{math.sqrt(running_loss / (i+1))}')

    # Validation loop
    model_v2.eval()
    total_val_loss = 0
    num_batches = 0
    rmse_sum = 0.0

    with torch.no_grad():
```

```
        for i, data in enumerate(test_loader, 0):
            input_ids = data['input_ids'].to(mps_device)
            attention_mask = data['attention_mask'].to(mps_device)
            time_series = data['time_series'].to(mps_device)
            targets = data['targets'].to(mps_device)

            outputs = model_v2(input_ids, attention_mask, time_series)
            min_price, max_price = outputs
            outputs_stacked = torch.cat((min_price, max_price), dim=1)
            loss = criterion(outputs_stacked, targets)

            total_val_loss += loss.item()
            num_batches += 1

        # Print validation loss
        print(f'Epoch [{epoch + 1}/{EPOCHS}], Validation Loss: {total_val_loss / num_batches}, RMSE:
{math.sqrt(total_val_loss / num_batches)}')

    val_loss = total_val_loss / num_batches
    if abs(val_loss - training_loss) < 0.01 and training_loss < 0.035:
      break

print('Finished Training')

# Evaluate the model on the test set and visualize the predictions
import matplotlib.pyplot as plt
model_v2.eval()  # set the model to evaluation mode
test_predictions = []
actual_values = []

with torch.no_grad():
    for data in test_loader:
        input_ids = data['input_ids'].to(mps_device)
        attention_mask = data['attention_mask'].to(mps_device)
        time_series = data['time_series'].to(mps_device)
        targets = data['targets'].to(mps_device)

        min_price, max_price = model_v2(input_ids, attention_mask, time_series)
```

```
        test_predictions.append(torch.cat((min_price, max_price), dim=1).cpu().numpy())
        actual_values.append(targets.cpu().numpy())
test_predictions_st_1 = [i * hour_future_price_min_std + hour_future_price_min_mean for i in
np.concatenate(test_predictions, axis=0)[:, 0]]
test_predictions_st_2 = [i * hour_future_price_max_std + hour_future_price_max_mean for i in
np.concatenate(test_predictions, axis=0)[:, 1]]

actual_values_st_1 = [i * hour_future_price_min_std + hour_future_price_min_mean for i in
np.concatenate(actual_values, axis=0)[:, 0]]
actual_values_st_2 = [i * hour_future_price_max_std + hour_future_price_max_mean for i in
np.concatenate(actual_values, axis=0)[:, 1]]

print(test_predictions_st_1)
print(test_predictions_st_2)

# Visualizing a sample of predictions
plt.figure(figsize=(15, 5))

# Min Price Predictions
plt.subplot(1, 2, 1)
plt.plot(actual_values_st_1[:], label="Actual Min Prices", color="blue")
plt.plot(test_predictions_st_1[:], label="Predicted Min Prices", color="red", linestyle="--")
plt.legend()
plt.title("Actual vs. Predicted Min Prices")



# Max Price Predictions
plt.subplot(1, 2, 2)
plt.plot(actual_values_st_2[:], label="Actual Max Prices", color="blue")
plt.plot(test_predictions_st_2[:], label="Predicted Max Prices", color="red", linestyle="--")
plt.legend()
plt.title("Actual vs. Predicted Max Prices")

plt.tight_layout()
plt.show()
```

ДОДАТОК Б ПРЕЗЕНТАЦІЯ